COMPUTER SCIENCE REVIEW XX (XXXX) XXX–XXX

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cosrev

Computer Science Review

Research paper

Sampling-based robot motion planning: Towards realistic applications

*Konstantinos I. Tsianos, Ioan A. Sucan, Lydia E. Kavraki**

Department of Computer Science, Rice University, Houston TX, USA

ARTICLE INFO

Article history:

ABSTRACT

This paper presents some of the recent improvements in sampling-based robot motion planning. Emphasis is placed on work that brings motion-planning algorithms closer to applicability in real environments. Methods that approach increasingly difficult motion-planning problems including kinodynamic motion planning and dynamic environments are discussed. The ultimate goal for such methods is to generate plans that can be executed with few modifications in a real robotics mobile platform.

1. Introduction

One of the important goals in robotics is to create a device – the robot – that can take as input a high-level specification of a simple task and execute it [39] without providing low level details on *how* to do so. An essential component of the task execution is for the robot to be able to move inside its environment. The latter typically requires the solution to a motion- planning problem, which has been one of the fundamental problems in robotics over the last couple of decades. Loosely stated, motion planning is the problem of deciding the set of motions that can take a robot from an initial to a final position while avoiding collisions [39]. Robots for planetary exploration, museum tour guides, search and rescue robots, robots in surgery are just a few out of the many examples of robotics applications that need motion planning [16,41]. Nowadays, motion planning is no longer restricted to just robotics applications. Structural analysis in biology [18] and computer graphics [22] are examples of developing research fields that can greatly benefit from the use of motion planning algorithms.

Depending on the type of robot, different difficulties need to be addressed by a motion-planning algorithm. Over the years, this has given rise to a number of directions in the field: planning for industrial manipulators [56,60,50, 59], mobile robots [42,12,48,6], humanoids [36], reconfigurable robots [1] are a few examples. This paper will focus on recent developments in motion planning that could eventually allow the use of motion-planning algorithms in real life applications for mobile robots. The principles of the developed algorithms apply, however, to the aforementioned types of robots as well.

A simplified version of the motion-planning problem is planning a collision-free path for a robot made of an arbitrary number of polyhedral bodies among an arbitrary number of polyhedral obstacles, between two collision free positions of the robot. Complexity analysis has shown this instance of the problem to be PSPACE-complete [54,13]. In cases where the problem is more complex (e.g. taking into account the physical properties, and actuator limitations in a real robot) it is not known if the problem is even decidable except for some particular cases [17].

* *Corresponding author.* Tel.: +1 713 348 5737; fax: +1 713 347 5390.
E-mail addresses: konstantinos@rice.edu (K.I. Tsianos), isucan@rice.edu (I.A. Sucan), kavraki@rice.edu (L.E. Kavraki).

1574-0137/$ - see front matter © 2007 Published by Elsevier Ltd
doi:10.1016/j.cosrev.2007.08.002

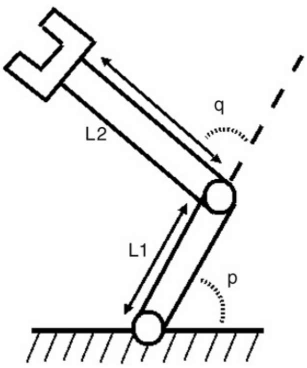

Fig. 1 – A configuration of the robot is completely specified by the values p and q.

Some of the well-known complete motion-planning algorithms are cell decomposition and visibility roadmaps [16,41]. For practical purposes, complete algorithms turn out to be computationally expensive and hard to implement. Adding various restrictions to the problem made the use of complete algorithms [28,23] possible. For the general case of the problem, a breakthrough was achieved with the development of *sampling-based motion planners* [7,38]. These algorithms quickly became popular for various reasons. Many previously considered hard problems could be solved using sampling-based motion planners, while the fundamental ideas behind these planners were in general easy to describe and implement. The increased performance of these algorithms comes at the cost of relinquishing completeness. Those algorithms can only guarantee *probabilistic completeness* instead. A probabilistically complete algorithm will eventually find a solution if there is one [35], but it will run forever if no solution exists.

In recent years, a number of review papers [46,14] have discussed issues in motion planning. This paper attempts to continue the work and present recent developments in the area of sampling-based motion-planning algorithms. The focus is on developments that may allow the application of sampling-based motion-planning algorithms on real mobile robots. Section 2 contains a formal description of the basic motion-planning problem. Section 3 presents motion-planning algorithms following the classic distinction into roadmap-based and tree-based planners. This section covers mostly algorithmic improvements in the fundamental modules that are present in most motion planners. Section 4 takes the ideas of the previous section one step further. New classes of motion-planning problems are introduced: problems that involve several realistic extensions to the basic problem. Finally, Section 5 summarizes the ideas described in the paper and discusses the future of sampling-based motion planners.

2. The motion-planning problem

As mentioned in the introduction, the motion-planning problem poses the question of how a robot can move from an initial to a final position. To acquire a formal statement of the problem, the position of the robot needs to be defined. A notion that has proved itself useful is that of a *configuration* (see Fig. 1). A configuration is a complete specification of the position of all the points on the robot. The set of all configurations forms the *configuration space*, $\mathcal{C}$. A robot described by a configuration is just a point in $\mathcal{C}$. The set of all configurations in $\mathcal{C}$ in which the robot is in collision with some obstacle in the environment is denoted by $\mathcal{C}_{\text{obst}}$. Similarly, the free space is defined as $\mathcal{C}_{\text{free}} = \mathcal{C} - \mathcal{C}_{\text{obst}}$. The motion-planning problem can be stated as follows:

Definition. Given an initial and a goal configuration q_{start}, $q_{\text{goal}} \in \mathcal{C}_{\text{free}}$, find a continuous path $p : [0, 1] \rightarrow \mathcal{C}_{\text{free}}$ where $p(0) = q_{\text{start}}$ and $p(1) = q_{\text{goal}}$.

This is the geometrical version of the motion-planning problem. The result is a collision-free path. This is usually known as *path planning*, since the planning algorithm is only asked to return a path, without considering the robot's ability to implement that path. This is not necessarily an issue if a robot is moving slow enough and the dynamic constraints such as friction, gravity, etc. can safely be ignored. As will be discussed in this paper though, there is an increasing interest in planning problems where the dynamic constraints can no longer be ignored. For those cases, planning algorithms need to come up not only with a geometrical path, but rather with what is called a *motion plan*, i.e. a complete description of what controls need to be applied so the robot can execute a feasible and collision-free trajectory to its goal. The term *motion planning* will be used in this paper, and it will be clear from the context whether the dynamic constraints are considered or not.

3. Recent improvements in sampling-based motion planning

Over the last few years, there has been a lot of work in improving sampling-based motion-planning algorithms. It is hard to define a single criterion that can classify all planners in distinct categories. The classical separation is between *roadmap-based* planners and *tree-based planners*. This section introduces the basic ideas present in almost all sampling-based motion planners and describes improvements in the aforementioned two categories of algorithms. Algorithms that deal with problems beyond purely geometrical path planning are presented in Section 4.

3.1. *Roadmap-based planners*

Roadmap-based planners are typically used as multi-query planners. As their name implies, they maintain a roadmap that can be used to answer different planning queries. The main data structure being used is a graph whose nodes are points in the configuration space. Edges in this graph exist between configurations that are close to one another, and the robot can move from one point to the other without collisions. A typical algorithm has two phases: a learning phase and a querying phase. In the learning phase, the roadmap is created:

- *Sampling*. Pseudo-random collision-free configurations called samples are generated. These are the vertices of the roadmap.

Please cite this article in press as: K.I. Tsianos, et al., Sampling-based robot motion planning: Towards realistic applications, Computer Science Review (2007), doi:10.1016/j.cosrev.2007.08.002

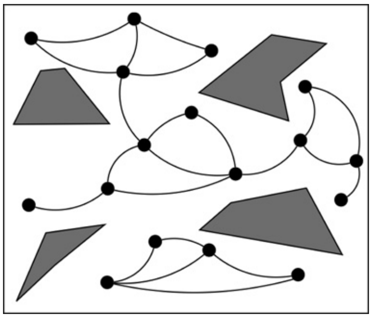

Fig. 2 – Sample roadmap.

- *Connecting*. A number of attempts are made to connect each sample to its nearest neighbours, thus adding edges to the roadmap.

To solve a particular query, the start and goal configurations are added to the roadmap and a graph search algorithm is used to find a path. The efficiency of the algorithm depends on how well the roadmap can capture the connectivity of the configuration space. Moreover, the main performance bottleneck is the construction of the roadmap, since graph search algorithms are fast.

Algorithm 1 presents the well-known PRM [38] method. Fig. 2 shows a sample roadmap with $k = 2$, where k stands for number of neighbours each sample tries to connect to. Two of the most important challenges of this method are how to sample useful configurations that will increase the coverage of the roadmap and how to connect samples in the roadmap.

Algorithm 1 BUILDROADMAP(k)

```
V ← {}, E ← {}
loop
  c ← SampleValidConfiguration()
  V ← V ∪ {c}
  N_k ← NearestNeighbours(V, k)
  for all n ∈ N_k do
    E ← E ∪ {(c, n)}
```

3.1.1. *Improving the sampling strategy*

In a sampling-based motion planner, one of the core issues is the *sampling strategy*. Sampling is the process by which new configurations are randomly selected to be added to the roadmap.

There are multiple possible directions for improving sampling. Some of the previous work focuses on sampling important areas of the configuration space using workspace information to derive what the important areas are. A well-known example is sampling in the areas of narrow passages [24,31]. Increasing the density of sampling around narrow passages increases the chances of finding samples in areas that are hard to reach and are likely to be needed for finding a solution. As an example, the bridge-test, presented in [24], uses information from samples found in collision in the following manner: if two samples x and $\tilde{x}$ are found in collision, their midpoint x_m (sample between x and $\tilde{x}$) is considered. If x_m is not in collision, it is added to the sample set.

Different sampling strategies have different strengths. For example, the bridge test described above is effective for sampling narrow passages. A fruitful idea was to try and combine the usually complimentary strengths of different sampling strategies. In [27] an adaptive strategy for selecting the most cost effective sampler out of a set of already existing ones is presented. The selection depends on the sampled region of the configuration space. Another idea is applying existing samplers in a chain-like fashion [58]. The starting sampler is always a uniform one; the following samplers take a sample as input and produce another one as output; a chain is formed by having the output of one sampler be the input of the next sampler. This combination yields good results for some sets of samplers in the sense that it combines the advantages of multiple samplers into one. The disadvantage of this idea is the increased overhead for generating samples. In [55], after an initial step of uniform sampling, the space is divided into regions. Depending on whether most samples were collision free or in collision, different regions are assigned different region-specific samplers. The region specific samplers are then used to further sample in a more cost effective way.

Another direction along the same lines is presented in [32]. The authors use different samplers for different components of the robot, where different components here refers to specific features of the robot geometry. The intuition behind this is that a solution – a path in the configuration space – corresponds to a path for every point on the robot in the workspace. The workspace is typically easier to reason about since a complete representation of it is available. Sampling according to certain features of the robot in the workspace produces different samplers. Information from these samplers is then used to guide the sampling process in the configuration space. The importance given to each of the feature samplers is dynamically updated using machine learning techniques.

3.1.2. *Improving the connection strategy*

In this section, some of the issues related to connecting samples in the *roadmap* are presented. While it may seem the more samples are connected, the better, connecting samples is a time consuming process and so a balance between number of connections and runtime needs to be achieved.

From a performance point of view, the main drawback of PRM is that it heavily relies on collision checking. To mitigate this effect, algorithms like Lazy PRM [5] have been designed. Lazy PRM delays collision checks by assuming edges to be valid and actually checking them only if they are part of potential solutions. To reduce the number of collision checks even further, and achieve better coverage of the configuration space at the same time, the use of predictive models has been introduced [2]. The idea behind predictive models is to compute an approximation of the configuration space using machine learning techniques. The approximation allows inferring the probability of a certain configuration being collision free. Use of these probabilities is made instead of collision checking when connecting samples in the roadmap. When a potential solution is found, edges

Please cite this article in press as: K.I. Tsianos, et al., Sampling-based robot motion planning: Towards realistic applications, Computer Science Review (2007), doi:10.1016/j.cosrev.2007.08.002

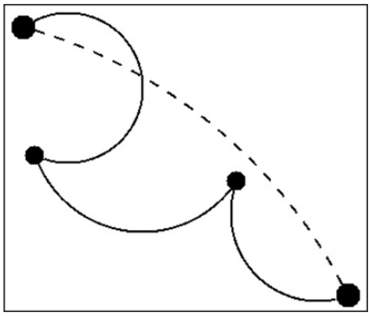

Fig. 3 – Nonoptimal solution found by motion planner (continuous line), optimal solution (dashed line), in an obstacle-free environment.

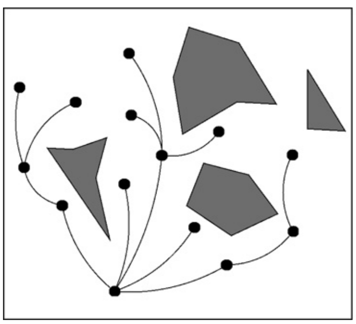

Fig. 4 – Sample RRT.

in the roadmap are validated using a collision checker. If a collision is found, samples around the end-points of the invalid edge are used in attempts to fix the roadmap. If fixing the roadmap fails, the roadmap building process is resumed until a path is found.

Solution paths obtained with a PRM planner are typically jagged and quite long (Fig. 3). Typically, a postprocessing – smoothing – step is applied to them. Even with this step, the produced path may still be far from the shortest one. In [49] a method for finding shorter paths has been presented. To allow PRM to find shorter paths in a reasonable amount of time, the connection strategy is changed so it allows cycles in the roadmap, with the condition that the edge that produces the cycle shortens the minimal path between the configurations it connects. Testing whether the connection criterion is met is done using a modified Dijkstra's algorithm. The modification speeds up the algorithm using the fact that the length of the minimal path is not needed — only it being longer or shorter than the new potential edge is the required information.

These connection strategies, while they do improve the planning algorithms, are specific for the path-planning problem. It is possible however, to use similar ideas in a motion-planning framework, as will be presented in Section 4.

3.2. *Tree-based planners*

In many cases, quickly solving one particular planning problem instance is of interest. For these cases, single query planners can be used. In these planners, the main data structure is typically a tree. The basic idea is that an initial sample (the starting configuration) is the root of the tree and newly produced samples are then connected to samples already existing in the tree. Significant amounts of work have been dedicated to developing sampling and connection strategies, biasing the direction in which the tree grows and achieving better coverage of the space. The most popular representative of tree-based planners is the RRT algorithm [40,44] (see also Fig. 4 and Algorithm 2). Many of the algorithmic improvements discussed in this section are using an RRT-like algorithm as a base. There are other tree-based planners though: EST [26], SBL [57], utility trees [3], a multiresolution version of [8] algorithm in [47], PDST [45], SRT [51] are well-known tree-based planners. Due to space limitations, not all of these algorithms are presented, but the reader is encouraged to look at the cited papers for details.

3.2.1. *Improvements in the RRT family of planners*

In the following paragraphs RRT and improvements to RRT like DDRRT [61] or AD-RRT [30] will be presented.

Algorithm 2 BUILDRRT(x_{init})

INIT(T, x_{init})
for $k = 1$ **to** $N_{\text{iterations}}$ **do**
  $x_{\text{rand}} \leftarrow$ RANDOMCONFIGURATION()
  $x_{\text{near}} \leftarrow$ NEARESTNEIGHBOUR(x_{rand}, T)
  if $x_{\text{new}} \leftarrow$ NEWSTATE($x_{\text{near}}, x_{\text{rand}}$) **then**
    INSERT(T, x_{new})
return T

The RRT algorithm works by growing a tree starting from a given root. The growth is performed one vertex at a time, by alternating the two steps that are common to most tree-based planners: *selection* and *propagation*.

- *Selection*
  - A sample x_{rand} is chosen uniformly at random.
  - Among the samples already existing in the tree, the closest one to x_{rand} is selected. Let this sample be x_{near}.
- *Propagation*
  - An edge is then extended from x_{near} toward x_{rand}, not necessarily reaching it.
  - The ending vertex from the edge extended from x_{near} is then the new sample added to the tree.

One of the bottlenecks of RRTs is that in some environments (see Fig. 5 for an example) most of the randomly selected samples will cause the expansion from the closest node in the RRT tree to fail. This produces a significant increase in the runtime of the algorithm. One way to mitigate this problem is presented in [61].

The idea is to attach a radius to each of the samples in the built tree. If the randomly-selected sample is further away than the specified radius, another sample is picked until the distance to the nearest sample in the tree is less than the attached radius. This change reduces the likelihood of having a connection failure. Samples added to the tree are initially set to infinite radius; when a connection attempt fails from a sample, its radius is set to some workspace-dependent constant.

An obvious issue with the method above is the workspace-dependent constant. This issue is addressed in [30]. Their idea

Please cite this article in press as: K.I. Tsianos, et al., Sampling-based robot motion planning: Towards realistic applications, Computer Science Review (2007), doi:10.1016/j.cosrev.2007.08.002

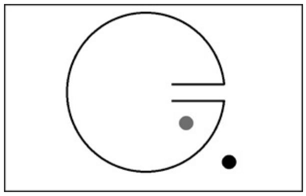

Fig. 5 – Bug trap. Starting point is inside the trap, goal is outside. Most of the random samples will be outside the trap and will fail to produce paths that exit it.

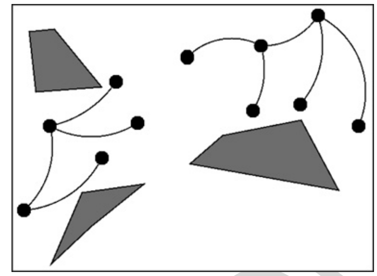

Fig. 6 – Two trees growing one towards the other.

is to adapt the value of the radius according to some other constant that is less sensitive to the workspace. The radius is increased with every successful connection attempt and decreased with every connection failure.

One of the newer RRT-like algorithms is based on utility trees [3]. The main improvement for this type of trees is that more aspects of the tree growth are evaluated: the utility of the node to be expanded, the expansion direction, the expansion distance and connection attempts. The utilities of the different aspects are evaluated using approximation techniques similar to those of predictive models presented above.

3.2.2. *Using multiple trees*

When solving a motion-planning problem, it is often the case that multiple trees are used. So-called bidirectional algorithms [33] grow trees both from the start and from the goal regions, one towards the other and try to connect them (Fig. 6). Another situation in which multiple trees are used is in algorithms like SRT [51]. The idea behind SRT is to have a roadmap of trees. Instead of connecting samples, trees are grown from each sample and they are connected to other nearby trees to form a roadmap. This is a generalization of roadmap-based and tree-based planners. The main advantage of using multiple trees is the potential for parallel execution.

An important issue that arises with algorithms that use multiple trees is the connection of the trees. Deciding which nodes in which trees need to be connected is not a simple issue. In addition, connecting two nodes is also a difficult problem in the context of *motion* planning. More details about how to deal with these difficulties follow in Section 4.1.

4. New directions in sampling-based motion planning

So far a number of different ideas that try to improve on the essential components present in sampling-based motion planners have been described. Most of the algorithms in the previous section have the underlying assumption that the robot is a free-flying 3-dimensional body moving in a static workspace. In the area of mobile robotics though, it is an interesting and challenging goal to try and embed a sampling-based motion planner in a real robot as a black box, that can automatically drive a robot to wherever its goal might be. For such functionality in real-life scenarios, there are various constraints and difficulties that need to be addressed on top of the basic geometrical motion-planning problem. This section tries to identify some of those issues, and show how sampling-based planners are being adapted to deal with them.

The extensions to the basic motion-planning problem that will be discussed are summarized below. For real robots these are not the only issues that need to be considered. Dealing with uncertainty in motion and sensors and consequently problems in localization and mapping, are very important but are omitted in this paper.

- **Robot's dynamics:** One crucial extension towards more physical realism is to try and take into account dynamic constraints. A real robot is not a "free-flying" object. It has motor limitations that impose bounds on its maximum velocity and acceleration [9,44]. These are called *kinodynamic constraints* and can significantly increase the complexity of motion planning, as the robot might be incapable of implementing certain collision-free paths (infeasible). Furthermore, real robots are subject to other physics-based constraints such as gravity, and friction [45] that can and sometimes need to be taken into account.
- **Workspaces that change in time:** Another extension is to relax the static workspace assumption. This is another important extension that is necessary for robots that are not restricted to operating in a highly-controlled, stationary environment. The difficulty motion planning in such cases can vary based on what is known about the moving obstacles. In the best case, the obstacles are executing repetitive motions and information about their maximum velocity or acceleration is available [11]. It could be though that the moving obstacles are unpredictable or even malevolent and moving arbitrarily fast. In these cases guaranteeing collision avoidance overall for a robot may be impossible [52,10].
- **Real-time planning:** In real life scenarios, it is frequently the case that a robot will need to move in an only partially known environment. In those cases, as new sensory information is obtained, the robot needs to be able to revise its plan, i.e. to *replan* [21]. Moreover, in environments that are changing in time, the robot is expected to react to these changes and replan in real-time while moving. Finally, all these considerations become more important when the robot's dynamics are taken also into account [6,12,20].

Please cite this article in press as: K.I. Tsianos, et al., Sampling-based robot motion planning: Towards realistic applications, Computer Science Review (2007), doi:10.1016/j.cosrev.2007.08.002

4.1. *Kinodynamic planning and physics based constraints*

Real robots have kinodynamic constraints that cannot generally be ignored. One common way of taking those constraints into account is with the use of an appropriate controller that can generate feasible motions. A very common approach to solve motion-planning problems is with a decoupled approach (decoupled trajectory planning) [37,12]. First, a path-planning algorithm computes a collision-free trajectory ignoring system dynamics. Then, a controller is needed to compute appropriate controls that will implement the desired path. There is a number of issues in this approach. Typically, controllers alone cannot avoid obstacles in the environment, and that is why an obstacle free path must be found in another way first. Moreover, the produced geometrical paths may be infeasible for a real robot and even when the controller manages to follow a desired path, this may require that the robot moves slowly to minimize the influence of dynamic and physical constraints. Finally, controllers are system specific, and as today's robots become increasing complex it becomes very hard to develop good controllers.

In the last few years a number of sampling-based motion planners and especially tree-based planners, have made it possible to accommodate kinodynamic constraints and physics constraints in a computationally feasible way. Sampling-based planners have a more unified approach as they produce feasible paths that at the same time avoid obstacles. Moreover, for a those planners also provide the time sequence of controls that the robots needs to execute to move on the selected path. The main idea behind sampling-based motion planners for kinodynamic planning, is to search a higher dimensional *state space* X that captures the dynamics of the system. Given a configuration $q \in C$, a state of a robot can be simply defined as $x = (q, \dot{q})$ [44]. The goal is to plan in the state space similarly to planning in the configuration space. In this way the techniques described in previous sections can be adapted to deal this new class of problems. In the first subsection, planners that are derived from classical tree-based planners such as RRT and EST are covered. Next, a new family of sampling-based motion planners called *path directed tree planners* is described. Last, some ideas are presented on how to use bidirectional trees in the presence of kinodynamic constraints.

4.1.1. *Classical tree-based planners*

The first fruitful attempts to incorporate kinodynamic constraints in a sampling-based planner, were based on modifying existing tree-based planners.

In [44] an RRT-like planner is described. The paper explains how dynamics can be incorporated in a sampling-based planning framework. The RRT-tree is produced in a way similar to what was described in Section 3. The difference is that here the planner samples random controls and tries to apply them for some amount of time in order to expand from a current state on the towards the newly sampled state. In this way, any path on the tree is a feasible and collision-free trajectory of the robot. The authors consider complex systems such as hovercrafts and satellites in environments that are cluttered with obstacles.

A similar way of planning under kinodynamic constraints is presented in [25]. Planning is done in the *state space* × *time* space in a fashion that follows another popular tree-based planner, the EST [26]. The planner picks a state node already on the tree and samples a random control that is applied for some amount of time to add a new node on the tree. The node for the next expansion is selected in a way so as to create a tree that is not too dense in some parts and too sparse in others. The authors provide an analysis of the probabilistic completeness of their algorithm. Moreover, they present experiments on non-holonomic robots both in simulation and for real robots. Some interesting ideas are also discussed with respect to recomputing a trajectory if there is an unexpected change in the environment that conflicts with the current trajectory.

Along the same lines is [20], which tries to show the decoupling between the higher level motion planner and the lower level control. Their approach is closer to RRTs but has some important differences. A state is chosen at random, and the planner tries to expand a tree towards a new sample. Yet, contrary to RRT, which is expanding from the node on the tree that is closest to the new sample, the authors evaluate the nodes of the current tree in order of increasing cost to the new sample using some distance metric. Expansion towards the new sample is attempted from all nodes on the tree before the sample is considered unreachable. An optimal control policy in the obstacle-free case is used to drive the robot. Moreover, this algorithm contains ideas about how to deal with real-time planning where the planner only has a time budget to produce a trajectory to the goal.

In all of the planners presented above, one of the issues that is dealt with is the direction of growth for the tree. On one hand, coverage needs to be eventually achieved in order to guarantee probabilistic completeness, on the other hand, goal bias needs to be taken into account, in order to speed planning. DSLX [53] (Discrete Search Leading Continuous eXploration) is proposed as a method to address this issue. The idea is that the workspace is discretized and a discrete path from start to goal is found. This path will be used as a hint, to lead the direction of growth of the tree. This method achieves significant computational improvements.

4.1.2. *Path-directed planners*

Most sampling-based planners require a distance metric in the space that is being sampled. Metrics are typically required for biasing the search and finding nearest neighbours to compute edges in the tree or roadmap. However, especially in state spaces, it can be hard and counter-intuitive to define a good metric between states. Moreover, metrics are usually not general enough and work well only for a specific system. The motivation for having a planner that does not depend on distance metrics lead in the last few years in the development of a new family of tree-based planners, called *path-directed planners*. The major difference of these planners is that the tree-data structure no longer uses single points as samples. Instead, the samples are whole-path segments that can hold useful information in order to speed the exploration of the planning space.

PDST [45] is the first planner in the family of path directed tree planners that introduced a new idea for creating a tree

Please cite this article in press as: K.I. Tsianos, et al., Sampling-based robot motion planning: Towards realistic applications, Computer Science Review (2007), doi:10.1016/j.cosrev.2007.08.002

which does not use a metric to bias the search. The basic scheme is illustrated in Algorithm 3. At each iteration, a sample γ is selected. Then a random state x on the selected sample is chosen and a new sample is propagated from that state by applying a newly randomly-selected control u for some time δt. The innovation is that PDST has a space-subdivision scheme and does not require a metric. The space is subdivided into cells. After a new sample is propagated, the cell in which that sample starts, is subdivided. An invariant of the algorithm is that each sample is contained only in one cell. The algorithm keeps track of how many samples are located in each space cell and can in this way estimate how dense the sampling is in different areas of the space. The selection of samples for expansion favours those that lead to new unexplored areas of the space. To guarantee probabilistic completeness, each sample also has an associated priority. Priorities are updated in a way that guarantees that eventually every sample in the tree will be selected for propagation.

PDST has been applied to a number of systems with complex dynamics, from cars and blimps to a weight lifting robot. An interesting idea, that has been tried is the combination of PDST with a physics engine, that simulates the world. In this way, the planner could be used to plan for systems even more realistic situations where physical constraints such as gravity and frictions are taken into account.

Algorithm 3 PDST(x_{init})

for $k = 1$ **to** $N_{\text{iterations}}$ **do**
  γ = SELECTSAMPLE()
  $(x, u, \delta t)$ = SELECTSTATECONTROLSURATION(γ)
  π = PROPAGATE($x, u, \delta t$)
  if INTERSECTGOALREGION(π) **then**
    TERMINATE()
  UPDATEPRIORITIES()
  SUBDIVIDE(GETCELLOF(γ))

Another path directed tree planner is [6]. This planner also uses a selection/propagation scheme to create new samples and generate a tree. This algorithm tries to avoid the overhead of subdivision while still not using a metric to bias the search. Instead, a low dimensional navigation function that has its global minimum in the goal region is defined. This navigation function can be computed very fast, and for any point in the workspace, it provides the A* distance of that point to the goal. Although this distance cannot capture the dynamics of the system, this work shows that it can bias the search to the goal sufficiently for simulated cars with second order dynamics. Assigning priorities to samples is used to guarantee probabilistic completeness.

4.1.3. *Path deformation and closing gaps*

The idea of using multiple trees exists in the case of kinodynamic motion planning as well. However, it not possible to analytically compute the controls needed for connecting nearby states and thus gaps may appear. In the following paragraphs, ideas of how to close such gaps are presented.

A possible option for closing the gaps is then perturbing the controls in the potential solution path into ones that achieve smaller gaps. Perturbing controls along such a path may require integration of potentially long sections of the path, which is time consuming. In [15], a method for replacing this integration with translation is presented. The method relies on using group symmetries in the system.

Another approach for closing gaps is using path deformation. The authors of [43] present a method of connecting two trees — one grown from the goal and one grown from the source. The method deforms the paths – one path starting at the source and the other ending at the goal – such that the free end-points of the two paths become closer and closer. This is an iterative process that aims to find a minimum using a potential field. The method may get stuck in local minima but experimental results show this rarely happens when attempting to connect reasonably close end-points.

4.1.4. *Remarks on kinodynamic planning*

The algorithms presented in this section show that the latest tree-based planners are becoming able to deal with kinodynamic constraints by planning directly in the state space. Trees are simple and efficient data structures that can represent temporal information in a natural way. Moreover, tree-based planners can overcome the difficulty that controllers face in implementing a desired path as the produced trajectories are always feasible.

The main problem that these algorithms tend to have is that the produced paths are generally suboptimal and typically contain cusps and sharp turns. Postprocessing and smoothing those paths is an active area of research that will not be covered in this paper.

4.2. *Dynamically-changing environments*

With the efficiency improvements of planners, interest has grown towards planning for robots in more realistic scenarios. For example, demand has emerged for planning amongst moving obstacles, dynamically-changing environments and/or unknown environments. In such cases, due to observed changes in the environment, the current plan can be rendered invalid and a new plan has to be produced. Moreover, time is an issue and the planner can only rely on temporarily valid information obtained from its sensors to quickly come up with a new motion plan while moving. These ideas are captured in the notions of real-time planning and replanning. Again, tree-based planners are proving to be a good framework that has been adjusted to deal with these kinds of problems. Nevertheless, there also exist some algorithms that use roadmaps.

4.2.1. *Basic replanning algorithms*

A simple replanning framework was presented in [21]. It presents an RRT planner that is the probabilistic analogue to the family of D* algorithms [34]. Specifically, an RRT tree is grown to cover the space until an obstacle is sensed in the way. The paper describes how part of the tree is quickly invalidated. The algorithm tries to expand towards the goal from what is left of the pruned tree. Along the same lines,

Please cite this article in press as: K.I. Tsianos, et al., Sampling-based robot motion planning: Towards realistic applications, Computer Science Review (2007), doi:10.1016/j.cosrev.2007.08.002

but in a PRM framework, is [4] which also tries to produce paths that optimize some criterion, such as time, stealth etc. In this work, the robot first builds a roadmap in the state × time space of the environment and finds an initial plan that takes any known dynamic obstacles into account. Then, as the robot starts executing the plan, it is possible that new obstacles might be observed that invalidate the plan. In that case, a discrete search algorithm called Anytime D*, is employed. This algorithm can quickly repair the plan, so it no longer interferes with the moving obstacles. The above ideas are closely related to Artificial Intelligence techniques, where replanning has been studied for longer time in a discrete graph search context.

4.2.2. *Planning amongst moving obstacles with roadmaps*

This subsection presents algorithms that dynamic environments into account with the use of roadmaps. The robot's dynamics is ignored so planning is done in the configuration space. For environments where obstacles are not necessarily static, a fixed roadmap cannot maintain information about the connectivity of the space. There are two main directions for addressing this problem. One assumes the movement of obstacles is predictable and then time can be considered an extra parameter of the configuration space. This basically allows using roadmap-based algorithms in a higher dimensional space. The other direction is to use roadmaps that permit updates. This is a more general method but raises the problem of updating the roadmap in a useful and efficient manner.

As an example of the first direction, planning in environments with obstacles that have known periodic motions has been examined in [11]. In order to improve efficiency, instead of simply adding a time component, to each point in the configuration space a period is associated — the interval at which the point is in collision. The points that do not change from $\mathcal{C}_{\text{free}}$ to $\mathcal{C}_{\text{obs}}$ have a period of zero. Compared to simply augmenting the configuration space with a time component, the presented method is more efficient.

For the second direction, ideas from [29,60] are presented. In [29], relevant portions of the roadmap are checked for collision with known dynamic obstacles for every query. A bidirectional tree-based planner is used to attempt restoration of the connectivity lost from edges that are in collision. If the tree-based algorithm fails, more samples are added to the roadmap. This allows the roadmap to be potentially updated with every query.

A similar notion is presented in [60]: samples in the roadmap are allowed to move and change their connectivity — an elastic roadmap. However, the connectivity here is defined by the ability of a feedback controller to move the robot between connected states. Another major difference is that the roadmap is no longer in the configuration space, but in the workspace. These changes allow faster computation for some problems but lose the probabilistic completeness property of the planner.

4.3. *Online replanning for robots with kinodynamic constraints*

In this subsection, two algorithms that are combining many of the ideas described above are presented. The robots considered have nontrivial kinodynamic constraints and they move in an environment that is partially known and/or changing. For this reason, robots have to gather new information periodically, and then replan using the latest available information. This is one of the most interesting and relatively newest classes of problems so the present literature is quite limited.

In [6], a tree-based planner for car-like robots with second order acceleration constraints is described. A robot is trying to explore an unknown environment. This work shows how previously computed trees can be reused efficiently in the next replanning step. More specifically, by retaining the valid part of a previously computed tree, the planner is able to avoid redundant collision checks. In many cases, the quality of the returned paths towards a chosen goal is improving in consecutive replanning steps. It is important to emphasize that this planner is computing plans in real-time under a time budget.

Another work that deals with robots that have nonholonomic kinodynamic constraints is presented in [42]. Here, an initial plan in the state space is computed with a sampling-based motion planner. Then, the robot starts executing that plan until it senses some change in the environment or deviation from the specified trajectory that renders the current plan invalid. At that point the robot has to replan. This is done by deforming the path in a way that still respects the nonholonomic constraints.

4.3.1. *Safety*

To close the section of new directions in sampling-based motion planning, it interesting to see how all the extensions to the basic motion-planning problem can coexist in a planning problem. A robot moving in an unknown and/or changing environment needs to change its plan rapidly, depending on the latest sensor input. Yet, if the robot is limited by its dynamic constraints, it cannot instantaneously change its behaviour. All of these considerations have brought up the issue of safety. It is no longer enough to simply produce feasible trajectories that are collision free with respect to static or moving obstacles. The trajectories have to also be *safe*. Safety has been defined in different ways in the literature, but a simple and generic description defines as safe a plan where the robot never finds itself in what is called an *Inevitable Collision State* or ICS [19]. Being in ICS means that due to dynamic constraints, the robot will collide with an obstacle in the future no matter what controls are applied from that state on.

One recent paper that incorporates many of the issues discussed here and in the previous section is [12]. This work deals with real robots that participate in the RoboCup competition. The robots move fast, so dynamics cannot be ignored; the environment is rapidly changing since there are many other robots (in the same or the opposing team) moving in the same area. The robots have a very small time budget to plan their next motion. This paper describes a three stage algorithm. First, an RRT-like planner finds a path to the desired goal position, ignoring dynamics. Then, a controller needs to find the appropriate controls that implement the path. There is also a third stage, responsible for producing safe paths. Out of the possible valid solutions, a search is

Please cite this article in press as: K.I. Tsianos, et al., Sampling-based robot motion planning: Towards realistic applications, Computer Science Review (2007), doi:10.1016/j.cosrev.2007.08.002

performed to filter out all solutions that can potentially lead to inevitable collisions in the future.

The notion of ICS is also used in [6], to define and guarantee the safety of an exploring robot. Specifically, the algorithm accepts only the trajectories for which after the last state of a trajectory, there exists a contingency plan. The contingency plan, describes a plan that the robot can always execute in that state, in order to avoid collisions in case the planner fails to produce (i.e. due to time limitations) any other safe trajectory to the goal.

5. Conclusion

Motion planning is an important problem in robotics and many approaches to solving it have been examined. Even though complete algorithms are PSPACE-complete and thus not useful for practical purposes, probabilistically complete algorithms have been very successful in a variety of problems. These algorithms form the category of *Sampling-Based Motion Planners*.

Sampling-based motion planners have been used to solve difficult geometrical problems, but have also proven flexible enough to deal with more realistic, hard, motion-planning problems. From the mobile robotics point of view, this work discussed planning for robots with kinodynamic constraints and planning in dynamic environments. A detail to note is most of the algorithms are not specifically designed for mobile robots. They are general and powerful algorithms that are also used in other areas or robotics such as manipulators, humanoids and reconfigurable robots. Due to space limitations, topics on these areas are not presented in this work.

While much progress has been made over the last decades, motion planning for real robots that can operate in everyday life scenarios, is still at its beginnings. Sampling-based motions planners started mainly as offline planners for geometric problems and static environments. Research in the last years has shown that such planners could be a powerful alternative in planning for real robots as well. However, there is still a number of issues that have to be addressed before installing a sampling-based motion planner on a real robot becomes possible. Real systems push current planners to their computational limits as the state space can be high dimensional. Moreover, planning in the state space is not fully understood or intuitive, as narrow passages (the main difficulty of sampling-based planners) can appear due to dynamic constraints. The quality of the paths produced by sampling-based motion planners is another problem and it is an active area of research. Finally, there is the issue of uncertainty in motion, which is inevitable is real systems, and is again an area of active research in the context of sampling-based motion planning.

Acknowledgments

This work has been supported in part by NS CNS 0615328 and IIS 0308237.

REFERENCES

[1] F. Aghili, K. Parsa, Configuration control and recalibration of a new reconfigurable robot, in: IEEE International Conference on Robotics and Automation, 2007, pp. 4077–4083.
[2] B. Burns, O. Brock, Sampling-based motion planning using predictive models, in: IEEE International Conference on Robotics and Automation, Barcelona, Spain, April 2005.
[3] B. Burns, O. Brock, Single-query motion planning with utility-guided random trees, in: IEEE International Conference on Robotics and Automation, Rome, Italy, 2007.
[4] J.v.d. Berg, D. Ferguson, J. Kufner, Anytime path planning and replanning in dynamic environments, in: IEEE International Conference on Robotics and Automation, 2006.
[5] R. Bohlin, L. Kavraki, Path planning using lazy prm, in: IEEE International Conference on Robotics and Automation, vol. 1, 24–28 April 2000, pp. 521–528.
[6] K.E. Bekris, L.E. Kavraki, Greedy but safe replanning under kinodynamic constraints, in: IEEE International Conference on Robotics and Automation, 2007.
[7] J. Barraquand, J.-C. Latombe, Robot motion planning: A distributed representation approach, International Journal of Robotics Research 10 (6) (1991) 628–649.
[8] J. Barraquand, J. Latombe, Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles, Algorithmica 10 (1993) 121–155.
[9] F. Boyer, F. Lamiraux, Trajectory deformation applied to kinodynamic motion planning for a realistic car model, in: IEEE International Conference on Robotics and Automation, 2006.
[10] J.P.v.d. Berg, M.H. Overmars, Computing shortest paths amidst growing discs in the plane, in: European Workshop on Computational Geometry, March 2006, pp. 59–62.
[11] J.P.v.d. Berg, M.H. Overmars, Path planning in repetitive environments, in: Methods and Models in Automation and Robotics, 2006, pp. 657–662.
[12] J. Bruce, M. Veloso, Real-time multi-robot motion planning with safe dynamics, in: Multi-Robot Systems: From Swarms to Intelligent Automata, volume III, 2006.
[13] J. Canny, Some algebraic and geometric computations in pspace, in: Annual ACM Symposium on Theory of Computing, Chicago, Illinois, United States, ACM Press, 1988, pp. 460–469.
[14] S. Carpin, Randomized motion planning — a tutorial, International Journal of Robotics and Automation 21 (3) (2006) 184–196.
[15] P. Cheng, E. Frazzoli, S. LaValle, Improving the performance of sampling-based planners by using a symmetry-exploiting gap reduction algorithm, in: IEEE International Conference on Robotics and Automation, vol. 5, 26 April–1 May 2004, pp. 4362–4368.
[16] H. Choset, K.M. Lynch, S. Hutchinson, G.A. Kantor, W. Burgard, L.E. Kavraki, S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, ISBN: 0-262-03327-5, June 2005.
[17] P. Cheng, G. Pappas, V. Kumar, Decidability of motion planning with differential constraints, in: IEEE International Conference on Robotics and Automation, 2007, pp. 1826–1831.
[18] J. Cortés, T. Siméon, V. Ruiz De Angulo, D. Guieysse, M. Remaud-Siméon, V. Tran, A path planning approach for computing large-amplitude motions of flexible molecules, Bioinformatics 21 (1) (2005) 116–125.
[19] T. Fraichard, H. Asama, Inevitable collision states: A step towards safer robots? in: Conference on Intelligent Robots and Systems, 2003.

Please cite this article in press as: K.I. Tsianos, et al., Sampling-based robot motion planning: Towards realistic applications, Computer Science Review (2007), doi:10.1016/j.cosrev.2007.08.002

[20] E. Frazzoli, M. Dahleh, E. Feron, Real-time motion planning for agile autonomous vehicles, in: American Control Conference, vol. 1, 2001, pp. 43–49.

[21] D. Ferguson, N. Kalra, A. Stentz, Replanning with RRTs, in: IEEE International Conference on Robotics and Automation, 2006.

[22] E. Ferre, J.-P. Laumond, An iterative diffusion algorithm for part disassembly, in: IEEE International Conference on Robotics and Automation, New Orleans, USA, 2004, pp. 3149–3154.

[23] S. Hirsch, D. Halperin, Hybrid motion planning: Coordinating two discs moving among polygonal obstacles in the plane, in: WAFR, Nice, 2002, pp. 225–241.

[24] D. Hsu, T. Jiang, J. Reif, Z. Sun, The bridge test for sampling narrow passages with probabilistic roadmap planners, in: IEEE International Conference on Robotics and Automation, 2003.

[25] D. Hsu, R. Kindel, J.-C. Latombe, S. Rock, Randomized kinodynamic motion planning with moving obstacles, 2000.

[26] D. Hsu, J.-C. Latombe, R. Motwani, Path planning in expansive configuration spaces, in: IEEE International Conference on Robotics and Automation, vol. 3, April 1997, pp. 2719–2726.

[27] D. Hsu, G. Sanchez-Ante, Z. Sun, Hybrid PRM sampling with a cost-sensitive adaptive strategy, in: IEEE International Conference on Robotics and Automation, 2005.

[28] D. Halperin, C.-K. Yap, Combinatorial complexity of translating a box in polyhedral 3-space, Computational Geometry: Theory and Applications 9 (1998) 181–196.

[29] L. Jaillet, T. Siméon, A PRM-based motion planner for dynamically changing environments, in: Intelligent Robots and Systems, Sendai, Japan, 2004.

[30] L. Jaillet, A. Yershova, S.M. LaValle, T. Siméon, Adaptive tuning of the sampling domain for dynamic-domain rrts, in: IEEE International Conference on Intelligent Robots and Systems, 2005.

[31] H. Kurniawati, D. Hsu, Workspace importance sampling for probabilistic roadmap planning, in: IEEE/RSJ International Conference on Intelligent Robots & Systems, 2004.

[32] H. Kurniawati, D. Hsu, Workspace-based connectivity oracle, an adaptive sampling strategy for prm planning, in: International Workshop on the Algorithmic Foundations of Robotics, 2006.

[33] J.J. Kuffner, S.M. LaValle, RRT-connect: An efficient approach to single-query path planning, in: IEEE International Conference on Robotics and Automation, 2000.

[34] S. Koening, M. Likhachev, Improved fast replanning for robot navigation in unknown terrain, in: IEEE International Conference on Robotics and Automation, 2002.

[35] L.E. Kavraki, J.-C. Latombe, R. Motwani, P. Raghavan, Randomized query processing in robot path planning, Journal of Computer and System Sciences 57 (1) (1998) 50–60.

[36] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, H. Inoue, Motion planning for humanoid robots, in: Proc. 20th Int'l Symp. Robotics Research, ISRR'03, 2003.

[37] M.B. Kobilarov, S.G.S., Near time-optimal constrained trajectory planning on outdoor terrain, in: IEEE International Conference on Robotics and Automation, 2005.

[38] L.E. Kavraki, P. Svestka, J.-C. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high dimensional configuration spaces, IEEE Transactions on Robotics and Automation 12 (4) (1996) 566–580.

[39] J. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Boston, MA, 1991.

[40] S.M. LaValle, Rapidly-exploring random trees: A new tool for path planning, Technical Report 11, Computer Science Dept., Iowa State University, 1998.

[41] S.M. LaValle, Planning Algorithms, Cambridge University Press, Cambridge, UK, 2006. Available at http://planning.cs.uiuc.edu/.

[42] F. Lamiraux, D. Bonnafous, Reactive trajectory deformation for nonholonomic systems: Application to mobile robots, in: IEEE International Conference on Robotics and Automation, vol. 3, Washington, DC, USA, 2002, pp. 3099–3104.

[43] F. Lamiraux, E. Ferr, E. Vallee, Kinodynamic motion planning: Connecting exploration trees using trajectory optimization methods, in: IEEE International Conference on Robotics and Automation, vol. 4, 2004, pp. 3987–3992.

[44] S.M. LaValle, J.J. Kuffner, Randomized kinodynamic planning, International Journal of Robotics Research 20 (5) (2001) 378–400.

[45] A.M. Ladd, L.E. Kavraki, Motion planning in the presence of drift, underactuation and discrete system changes, in: Robotics: Science and Systems, MIT Press, Boston, June 2005, pp. 233–241.

[46] S. Lindemann, S.M. LaValle, Current issues in sampling-based motion planning, in: Robotics Research: The Eleventh International Symposium, Springer-Verlag, Berlin, 2005, pp. 36–54.

[47] S. Lindemann, S.M. LaValle, A multiresolution approach for motion planning under differential constraints, in: IEEE International Conference on Robotics and Automation, 15–19 May, 2006, pp. 139–144.

[48] S.R. Lindemann, S.M. LaValle, Simple and efficient algorithms for computing smooth, collision-free feedback laws, International Journal of Robotics Research (2006).

[49] D. Nieuwenhuisen, M.H. Overmars, Useful cycles in probabilistic roadmap graphs, in: IEEE International Conference on Robotics and Automation, New Orleans, April 2004.

[50] A.L. Olsen, H.G. Petersen, Motion planning for gantry mounted manipulators: A ship-welding application example, in: IEEE International Conference on Robotics and Automation, 2007, pp. 4782–4786.

[51] E. Plaku, K. Bekris, B. Chen, A. Ladd, L. Kavraki, Sampling-based roadmap of trees for parallel motion planning, IEEE Transactions on Robotics 21 (4) (2005) 597–608.

[52] S. Petti, T. Fraichard, Safe motion planning in dynamic environments, 2005.

[53] E. Plaku, M.Y. Vardi, L.E. Kavraki, Discrete search leading continuous exploration for kinodynamic motion planning, in: Robotics: Science and Systems, Atlanta, Georgia, 2007.

[54] H.J. Reif, Complexity of the mover's problem and generalizations, in: IEEE Symposium on Foundations of Computer Science, 1979.

[55] S. Rodriguez, S. Thomas, R. Pearce, N.M. Amato, Resampl: A region-sensitive adaptive motion planner, in: International Workshop on the Algorithmic Foundations of Robotics, New York City, July 2006.

[56] G. Sanchez, J.-C. Latombe, On delaying collision checking in prm planning application to multi-robot coordination, International Journal of Robotics Research (2002).

[57] G. Sánchez, J.-C. Latombe, A single-query bi-directional probabilistic roadmap planner with lazy collision checking, International Journal of Robotics Research (2003) 403–407.

[58] S. Thomas, M. Morales, X. Tang, N.M. Amato, Biasing samplers to improve motion planning performance, in: IEEE International Conference on Robotics and Automation, Rome, Italy, April 2005.

[59] J. Vannoy, John Xiao, Real-time motion planning of multiple mobile manipulators with a common task objective in shared work environments, in: IEEE International Conference on Robotics and Automation, 2007, pp. 20–26.

[60] Y. Yang, O. Brock, Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation, in: Robotics: Science and Systems, 2006.

[61] A. Yershova, L. Jaillet, T. Siméon, S. LaValle, Dynamic-domain rrts: Efficient exploration by controlling the sampling domain, in: IEEE International Conference on Robotics and Automation, Barcelona, Spain, 2005.

Please cite this article in press as: K.I. Tsianos, et al., Sampling-based robot motion planning: Towards realistic applications, Computer Science Review (2007), doi:10.1016/j.cosrev.2007.08.002