Optimizing Robot Motion Strategies for Assembly with Stochastic Models of the Assembly Process

Rajeev Sharma
Steven M. LaValle
Seth Hutchinson

The Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
405 N. Mathews Avenue
Urbana, IL 61801

Abstract

Gross-motion planning for assembly is commonly considered as a distinct, isolated step between task sequencing/scheduling and fine-motion planning. In this paper we formulate a problem of delivering parts for assembly in a manner that integrates it with both the manufacturing process and the fine motions involved in the final assembly stages. One distinct characteristic of gross-motion planning for assembly is the prevalence of uncertainty involving time — in parts arrival, in request arrival, etc. We propose a stochastic representation of the assembly process, and design a state-feedback controller that optimizes the expected time that parts wait to be delivered. This leads to increased performance and a greater likelihood of stability in a manufacturing process. Six specific instances of the general framework are modeled and solved to yield optimal motion strategies for different robots operating under different assembly situations. Several extensions are also discussed.

Keywords: *assembly planning, motion planning, time-varying environment, stochastic uncertainty, numerical optimization*

This work was supported financially in part by the National Science Foundation under grant IRI-9216428 and Energy Power Research Institute (EPRI) under contract RP 8030-14.

1 Introduction

Modern manufacturing systems are confronted with planning problems at many scales, ranging from long-term production control, which deals with entire factories and time scales on the order of weeks, or even years, to fine-motion planning, which deals with individual robot assembly operations. Figure 1 illustrates a typical manufacturing system.

At the highest level, planning problems (usually called scheduling problems at this level) address the flow of parts through the assembly system. Production control determines what the assembly plant should be producing from, for example, month to month [6, 20, 39]. Given a set of production goals, the shop scheduler is given the task of determining how each robot cell will respond as parts arrive at its input buffers [5, 26, 29, 33]. For example, in Figure 1, workcell **RC5** receives parts from the three workcells **RC2**, **RC3**, and **RC6**, as well as some parts that are directly fed to **RC5** as input to the assembly system (i.e., parts P4, P7, P8, P9, P10 and P11). It is the task of the shop scheduler to determine the order in which workcell **RC5** will process these parts.

At a lower level, each workcell confronts a number of more specialized planning problems [11]. Figure 2 provides a more detailed, stylized view of an individual robot workcell. A sequence planner determines constraints on the order in which the robot will perform assembly operations [7, 14, 13, 16, 27, 42]. A gross-motion planner constructs the trajectories that the robot will execute in performing the tasks [23]. And, finally, a fine-motion planner determines robust, local strategies for the assembly operations, that are guaranteed to succeed, even in the presence of significant uncertainty [9, 10, 28]. Most often, as illustrated by the work cited above, each of these planning problems is treated in isolation.

In this paper, we take a first step toward integrating several levels of planning within a unified framework. In particular, we consider the problem of optimal gross-motion planning for a robot in an individual assembly cell, within the larger context of a full manufacturing environment. In the past, gross-motion planning has been treated as either a purely geometric problem (e.g., plan motion from point a to point b, avoiding collision with obstacles), or as an optimal control problem (e.g., find the time-optimal, or minimum energy path between point a and point b). In either case, the context in which the gross motion commands will be executed is ignored. In particular, motions are initiated at the request of a higher-level scheduling system, and at the end of the gross motion,

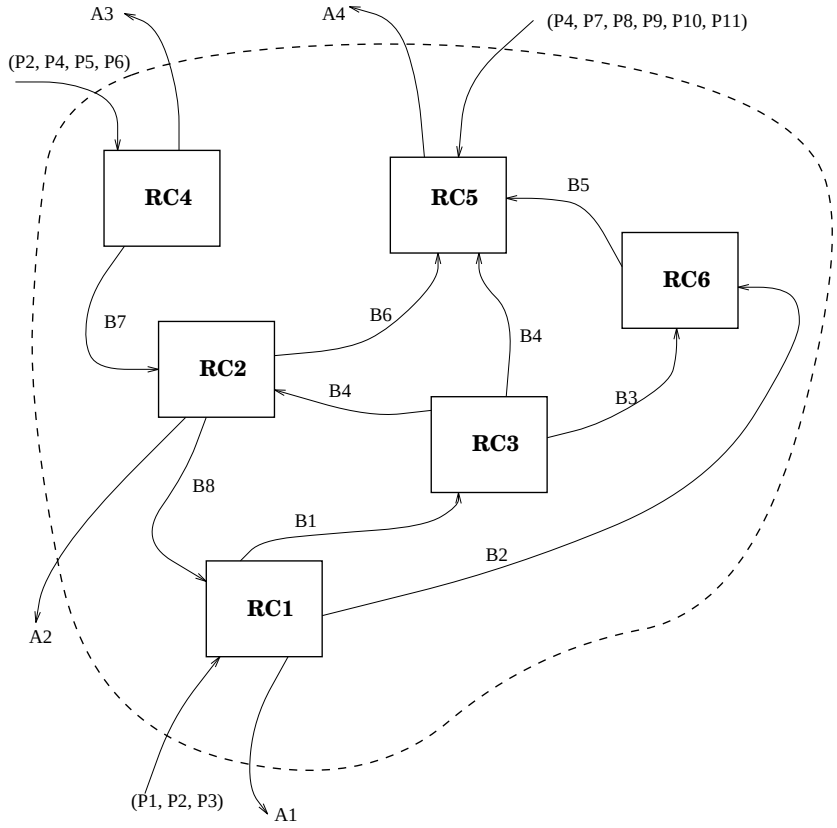


Figure 1: An assembly plant with multiple robot cells. P's are the parts, B's are the subassemblies, A's are the assemblies and RC's are the assembly robot workcells.

a fine motion assembly operation is performed.

If all aspects of the manufacturing system behaved deterministically, we could, in principal, treat gross-motion planning as a path optimization problem: derive the optimal path to move the parts from their initial positions to their destinations, on a schedule given *a priori* by the scheduler. However, real manufacturing systems are not deterministic. There is uncertainty in parts arrival time, position and orientation of parts to be manipulated, robot control, dimensions of manufactured parts, etc. Therefore, a reasonable goal of gross-motion planning would be to optimize the average or *expected* performance over time.

We characterize the problem of gross-motion planning for assembly as follows. A scheduler issues requests to the robot to grasp a particular part from a specified source, and to deliver the part to a specified destination. *A priori*, the only information regarding how these requests will be issued is in the form of a probability distribution on the set of possible part/source/destination requests. Because a fine-motion plan will often follow the execution of the gross motion, a source or destination is typically not specified as a single configuration, but is specified as a subset of the configuration space (which could in general be disconnected). The gross-motion planning problem is to derive a set of motion strategies that will produce optimal throughput of the assembly cell, in an expected sense.

Our gross-motion planning technique handles the stochastic nature of the assembly system by expanding the concept of planning in a configuration space that is combined with a Markov chain of modes. Each combination of part/source/destination request corresponds to a distinct *assembly mode*. In general each distinct manipulation that the robot performs (e.g., grasping a part, moving a part across the workcell) potentially changes the motion model or geometric model for the robot in its workcell. By using these concepts, we are able to optimize over a discrete set of possible state spaces, each corresponding to a unique combination of configuration space and assembly mode.

An important feature of our approach is the use of *motion strategies*. In classical geometric robot motion planning approaches, the output is usually a "motion plan" for a given description of the robot's configuration space, the initial and the goal positions. When unpredictable changes occur in the workcell, *dynamic replanning* is often used. This has been used, for example, in the context of error-detection and recovery [8], and task-level reasoning [12]. Alternatively, a fixed command might be given to the robot, and local collision avoidance would be performed to handle

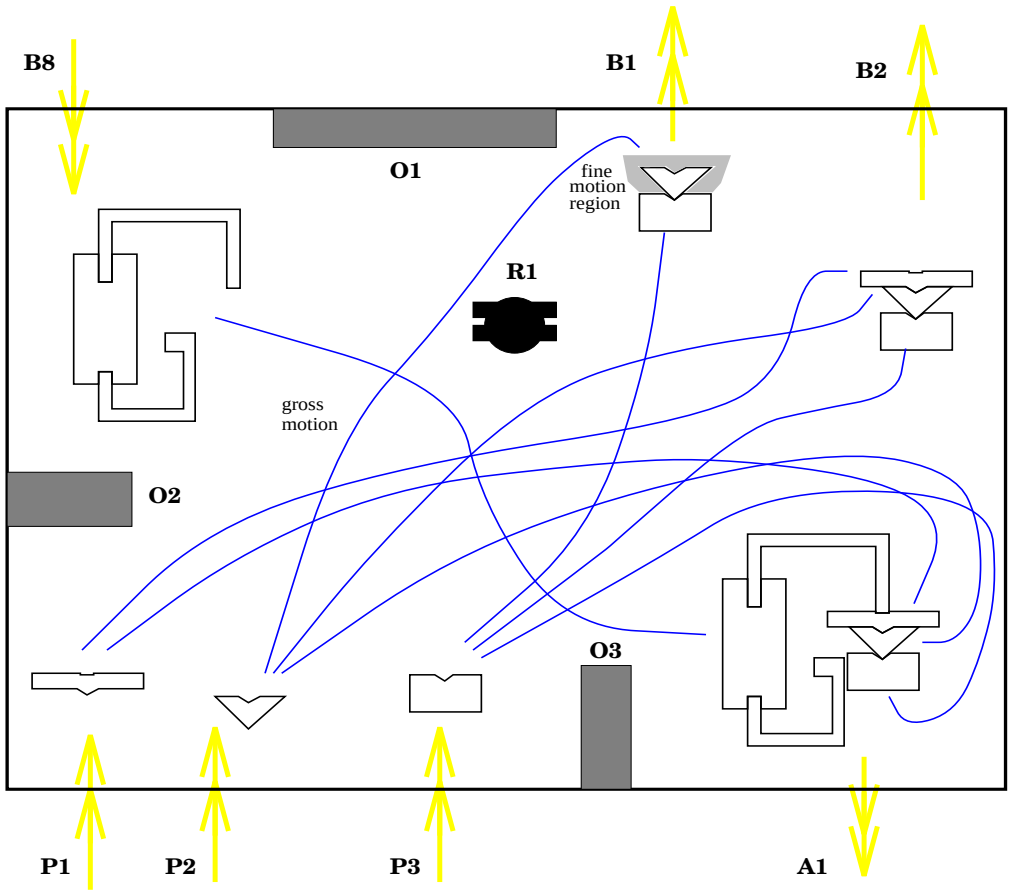


Figure 2: The motion planning problem in the robot workcell RC1, for multiple assembly from multiple components, with the robot R1 and the additional obstacles O1, O2, ans O3..

unexpected aspects of the environment [43]. In the probabilistic framework that we propose, a motion strategy provides a motion command for the robot for each contingency that might confront it. This motion strategy can be considered as a state-feedback stochastic controller [21] on a state space that simultaneously considers the assembly mode and the robot configuration. Replanning is not needed when the assembly mode changes, because the robot responds appropriately in different regions of the state space during execution. In addition, a state-feedback controller is advantageous since it will typically be robust to small modeling errors that can develop during execution [19, 4]. To select a motion strategy, we formulate an explicit performance criterion (or loss functional) that evaluates a trajectory executed by the robot. This allows a variety of factors, such as time, distance, or energy, to be optimized through the selection of a strategy.

The rest of the paper is organized as follows. Section 2 motivates the work by describing the specific features of gross-motion planning that are unique to assembly. Section 3 develops the mathematical model for the desired elements of assembly while Section 4 gives a computational scheme based on dynamic programming that determines optimal robot motion strategies. Section 5 presents a variety of specific assembly situations in which the model and computational technique is applied to obtain the optimal motion strategies. Section 6 discusses some applications and generalization of the technique to other assembly situations.

2 Background and Problem Description

In this section we consider those aspects of the gross-motion planning that are specific to the assembly situation. The discussion provides the basis for the models introduced in Section 3.

2.1 Gross-motion planning with changing geometry and motions

A robot is assumed to operate in a workcell, $\mathcal{W}$. In addition to static obstacles, the workcell contains a set of S *source regions*, denoted by $\{\mathcal{S}_1, \ldots, \mathcal{S}_S\}$, and D *destination regions*, denoted by $\{\mathcal{D}_1, \ldots, \mathcal{D}_D\}$. Let $\{\mathcal{P}_1, \ldots, \mathcal{P}_P\}$ denote a collection of P rigid parts. At various points in time, the robot will be requested to deliver some part from a source to destination. One important aspect in this problem is to establish the mapping from the workcell, $\mathcal{W}$, to the configuration space, C, of the robot. A standard gross-motion planning task is to determine a configuration-space path

whose image lies in the space of collision-free configurations, $\mathcal{C}_{free}$ [23].

When the robot is carrying a part or subassembly, the effective shape of the robot and load ensemble changes. Thus for each part $\mathcal{P}_i$ that the robot carries, the free configuration space is different. This concept of "changing" configuration space is an important aspect of the formulation of the motion planning problem as part of the assembly process. Related issues have been studied in the context of motion planning with movable obstacles [1, 41].

In addition to the geometric changes, the dynamics of the robot could change during the different stages of the assembly process due to the variation in loads [37]. Thus it is useful to incorporate the resulting changes into a motion planner. The models presented in Section 3 permit changes in the robot motion equation by allowing for different velocity commands for different parts to be manipulated.

2.2 Preconditions for fine-motion planning

The initial and final stages of moving a part for assembly involve *fine-motion planning*. During these stages the clearances between parts becomes significant relative to the uncertainties involved (e.g., [8, 9, 28]); hence sensing (e.g., force or torque sensing [17]) becomes an important part of the motion strategy. For gross-motion planning the usual approach is to ignore the fine-motion plan and consider the task of moving the robot between two points in its configuration space. Instead of defining a point-to-point motion goal, we allow the goals to be *regions* in the configuration space for both the grasp and ungrasp operations, based on relationships between the robot, the part, and the subassembly. Figure 3 shows an example of a source region and a destination region with respect to which appropriate initial and goal conditions for the gross-motion planning will be defined. In Section 3.3 we formalize these concepts to include generalizations such as disconnected source and destination regions, and fine-motion planning costs that depend on the configuration of the robot when it arrives in the region. This allows a better interface to be established between the gross-motion planning and the fine-motion planning for increasing the overall efficiency of the assembly process.

2.3 The concept of an assembly mode

The overall efficiency of a manufacturing facility can be improved if the gross-motion planning for assembly factors in more time-varying elements besides the ones considered so far. This includes, for example, the priorities and costs involved in the individual assembly motion subgoals. The priorities of a given operation in turn will be tied to the scheduling of the entire manufacturing facility as we will discuss in Section 2.4. For the gross-motion planning, thus, a useful concept for describing the current environment is an *assembly mode*. An assembly mode represents assembly information that is not part of standard gross-motion planning, including information that ties it to scheduling and fine-motion planning. The operation of the assembly robot can then be described in terms of a finite set of the assembly modes. The robot could cause a switch to new assembly mode by arriving at a particular region in its configuration space. We refer to a particular assembly mode in terms of the four-tuple: $\langle p, s, d, C/W \rangle$. This mode corresponds to the request that part P_p is to be transferred from source region S_s to destination D_d (see Figure 4). The fourth component indicates whether the part is being carried (C) by the robot or is waiting to be carried (W). We next discuss how the concept of assembly mode helps us define an important class of uncertainty in assembly involving time.

2.4 Scheduling and the assembly process

From the viewpoint of an individual workcell, the assembly process consists of a sequence of part/source/destination requests issued by a scheduler. These requests must be serviced by the robot according to a scheduling policy thereby inducing a sequence of assembly modes. Further, in each assembly mode a particular fine-motion strategy might be used to initially grasp the specified part, or to place the part in its goal position. In order for our new gross-motion planning approach to effectively optimize trajectories in an expected sense, the gross-motion planner must have some characterization of the anticipated behavior of the scheduling system in terms of the sequencing of requests and the fine-motion plans that must be executed for each request. In this section, we give a brief overview of scheduling methods, and describe how the resulting behavior of an individual workcell can be viewed as a Markov chain of assembly modes.

There have been many approaches to scheduling in large scale manufacturing systems. These range from heuristic methods for global optimization (e.g., [26]), to relaxation-based global methods

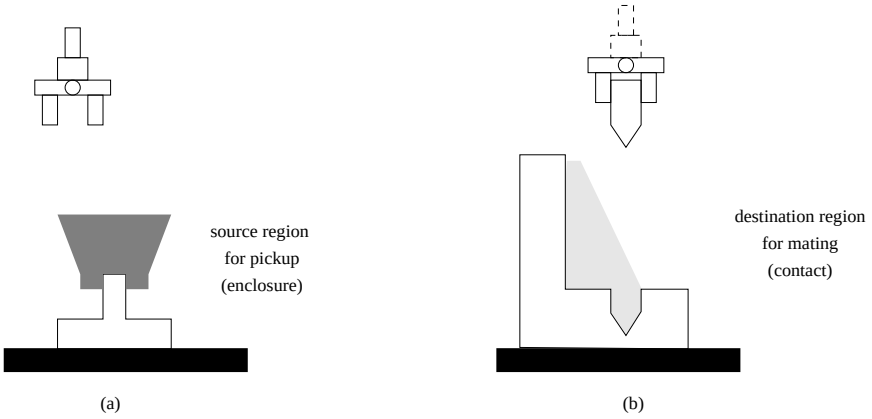


Figure 3: An example of (a) a source region and (b) a destination region, used for defining the gross-motion planning problem, while establishing good preconditions for fine-motion planning.

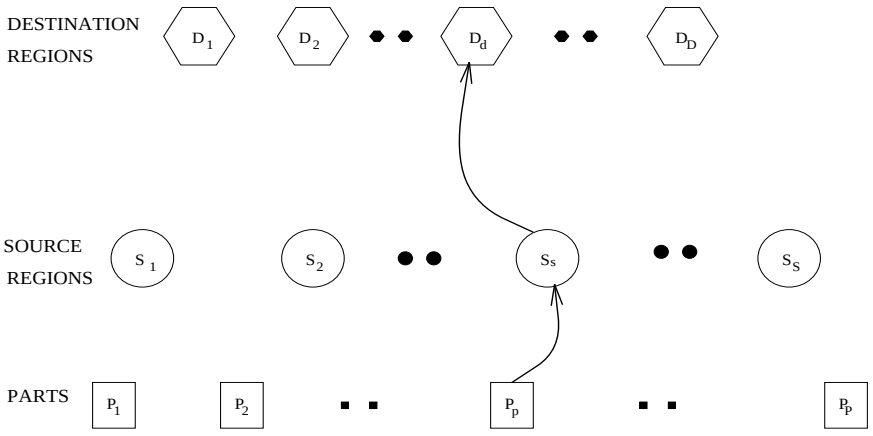


Figure 4: The abstract representation of the motion planning with the assembly in mode $\langle p, s, d, C/W \rangle$, the last component being C when the robot is carrying the part and W otherwise.

(e.g., [5]) to distributed, real-time scheduling policies (e.g., [33, 29]). In this paper, we will assume that a distributed, real-time scheduling approach is used, since such approaches scale to arbitrarily complex manufacturing systems, including job shops, flow shops, and re-entrant lines [33].

There are two primary concerns in designing a distributed scheduling system: *system stability* and *system performance*. If the system is not stable, the number of pending requests for a particular workcell may grow without bound, and as a result, the delay experienced by a part in the system may become unbounded. System performance is measured in terms of total throughput of the assembly system, both in terms of the mean delay experienced by a part in the system (cycle time) and the variance in the delay. The goal of a scheduling system is to optimize performance while ensuring overall system stability. One of the goals of our gross-motion planning system is to improve system performance (by increasing throughput), while preserving system stability, given a specific, stable scheduling policy.

The overall manufacturing system can be characterized as a stochastic process. In particular, the behavior of each individual workcell can be characterized as a random process that is conditioned on the behavior of a finite set of neighboring workcells. For example, in Figure 1, the behavior of **RC3** depends only on the output of **RC1**, while the behavior of **RC6** depends only on the output of **RC1** and **RC3**. We assume that a scheduler has been chosen that will lead to stability of the manufacturing system. Since the manufacturing system is a stochastic process, a scheduler will also behave stochastically. This in turn implies that the arrival of requests can be modeled as a stochastic process. The final process will be termed an *assembly process*, and is discussed in detail in the next section.

3 Mathematical Modeling

In this section we develop the mathematical concepts that model the gross-motion planning for assembly as discussed so far. Section 3.1 introduces the finite-state Markov process that is used to model the behavior of assembly modes, and the relationship of this process to the configuration space of the robot. In Section 3.2 defines state-feedback motion strategies. Section 3.3 introduces source and destination regions in the robot's configuration space that develop an explicit interface between the gross and fine motion planning for assembly through the definition of a performance

criterion.

3.1 Basic definitions

Let M denote the set of assembly modes, as discussed in Section 2.3. In addition to modes of the form $\langle p, s, d, C/W \rangle$, let $NR \in M$, denote a special mode that represents the condition in which no requests are to be processed. We assume here that the scheduling is done separately, and at a given time only one request can be waiting. Hence there are $2PSD + 1$ assembly modes. This choice was made to allow an external system to perform the scheduling, since it might be difficult or impossible for a robot acting in a workcell to make appropriate scheduling decisions that can affect an entire assembly process. This assumption is certainly not required, and Section 6 discusses the implications of considering multiple requests and allowing our system to alternatively perform the scheduling.

In general, to uniquely identify all of the possible situations that can occur in our problem, a *state space* is defined as a subset of the cartesian product, $X \subseteq \mathcal{C} \times M$. The assembly modes can be used to form a partition of the state space, X. Each time the assembly mode changes, the robot is forced into a different layer (or portion) of X. The set of free configurations varies across different layers. Let $M_w \subset M$ be the set of all modes such that a part is waiting to be picked up, and $M_c \subset M$ be the set of of all modes such that the robot is carrying a part. If $m = \langle p, s, d, W \rangle \in M_w$, then

$$\mathcal{C}^m_{free} = \{\mathbf{q} \in \mathcal{C} |\ \mathcal{A}(\mathbf{q}) \cap (\mathcal{B} \cup \mathcal{S}_1 \cup \cdots \mathcal{S}_{s-1} \cup \mathcal{S}_{s+1} \cdots \mathcal{S}_S) = \emptyset\}, \tag{1}$$

in which $\mathcal{A}(\mathbf{q})$ denotes the robot at configuration $\mathbf{q}$, and $\mathcal{B}$ denotes the static obstacle region (see [23]). In addition to avoiding collision with static obstacles, we also assume that the robot must avoid other source regions (this constraint can be removed for some applications).

Suppose $m = \langle p, s, d, C \rangle \in M_c$, which implies that the robot is carrying some part, $\mathcal{P}_p$. Let $\mathcal{P}(\mathbf{q})$ denote the transformed part, when grasped by the robot, which is at configuration $\mathbf{q}$. As discussed in Section 2.1, when a part is being carried by the robot, the effect is that of the "new" robot described as $\mathcal{A}(\mathbf{q}) \cup \mathcal{P}_p(\mathbf{q})$. Thus the free configuration space becomes

$$\mathcal{C}^m_{free} = \{\mathbf{q} \in \mathcal{C} |\ (\mathcal{A}(\mathbf{q}) \cup \mathcal{P}_p(\mathbf{q})) \cap (\mathcal{B} \cup \mathcal{S}_1 \cup \cdots \mathcal{S}_{s-1} \cup \mathcal{S}_{s+1} \cdots \mathcal{S}_S) = \emptyset\}. \tag{2}$$

The only remaining assembly mode on M is $m = NR$, in which

$$\mathcal{C}^m_{free} = \{\mathbf{q} \in \mathcal{C} |\ \mathcal{A}(\mathbf{q}) \cap (\mathcal{B} \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_S) = \emptyset\}. \tag{3}$$

3.2 Controlling the robot and the assembly process

In this subsection we describe how motion commands are given to a robot, and how these commands influence both the configuration of the robot and the assembly mode. These concepts lead to the definition of a strategy, which characterizes a given behavior that will be implemented in the robot.

We define a discretized representation of time by a set of *stages*, with an index $k \in \{1, 2, \ldots, K\}$, and stage k refers to time $(k-1)\Delta t$. We generally take Δt sufficiently small to approximate continuous trajectories. This appropriately reflects a situation in which a real robot is limited to some sampling rate for acquiring sensor information and executing motion commands. A final stage, K, is defined to preclude a special treatment of infinite stages, and in practice, $K\Delta t$ can be considered as the total time that the robot is in operation. An explicit choice of K does not need to be made, which will be discussed in Section 4. An *action* (or command), u_k, can be issued to the robot at each stage, k. We let U denote the *action space* for the robot, while requiring that $u_k \in U$. These actions can cause state transitions, which will be discussed shortly.

The behavior of assembly modes is modeled with a discrete-time Markov process, called the *assembly process*, which accounts for the uncertainty in future modes. At stage k, the probabilistic of the next assembly mode is given as $P(m_{k+1}|x_k, u_k)$, which generally depends on the current mode, m_k, current configuration, $\mathbf{q}_k$, and the current action, u_k. Although the probabilities can be chosen to model a wide variety of stochastic processes, we have derived the probabilities from a few basic transition types. In practice, any other Markovian model could be substituted in our approach without additional difficulty. We have considered the following three types of probabilistic state transitions:

1. The probability of receiving a p, s, d request while in mode NR

2. The probability that the destination will change to a new destination while in a carrying mode

3. The probability that the source will change to a new source while in a waiting mode

The first transition type is the most fundamental, and can be generally expressed as $P(m_{k+1}|m_k = NR) \geq 0$ if $m_{k+1} \in M_w$, and $P(m_{k+1}|m_k = NR) = 0$, otherwise. The second transition type can be expressed as $P(m_{k+1}|m_k \in M_c)$, which we allow to be nonzero only if m_k and m_{k+1} correspond to the same part and source. Ideally, the destination remains fixed, and $P(m_{k+1}|m_k \in M_c) = 1$ if $m_{k+1} = m_k$, and 0 otherwise. Similarly, the third type can be expressed as $P(m_{k+1}|m_k \in M_w)$, which we allow to be nonzero for any value of $m_{k+1} \in M$. Ideally, $P(m_{k+1}|m_k \in M_w) = 1$ if $m_{k+1} = m_k$, and 0 otherwise.

Any of these transition probabilities can be derived from an underlying Poisson process, which has been used extensively in the modeling of scheduling systems. The Poisson process is a reasonable choice for many problems because it captures several realistic properties: (i) The probabilities that arrivals occur in two non-overlapping time intervals are independent of each other; (ii) The probability that an arrival will occur in an interval is proportional to the length of the interval; (iii) The probability that an arrival will occur in an interval becomes arbitrarily small if the interval is made sufficiently small. Let λ denote a Poisson arrival rate. The density for the time of the first arrival is $p(t_a) = \lambda e^{-\lambda t_a}$. The probability that a transition will occur in time Δt is

$$P = \int_0^{\Delta t} \lambda e^{-\lambda t_a} dt_a = 1 - e^{-\lambda \Delta t}. \tag{4}$$

Thus, a choice of P results in an implicit choice of λ. In many manufacturing systems, stochastic models are assumed to be given. If this assumption is not made, however, values for λ can be estimated by merely observing the system for a period of time and collecting statistics. In particular, a Poisson frequency parameter, λ, can be estimated by counting the number of arrivals in the assembly system, and dividing by the period of time over which this counting occurs.

In addition to the three transitions listed earlier, there are several other key transitions that we model deterministically: from elements in M_w to elements in M_c, and from elements in M_c to NR. Suppose that the robot has an action, $FMP \in U$, that represents fine-motion planning. To grasp or ungrasp a part, the robot can choose this action from state x_k (causing the fine-motion operation to be performed), and the robot is returned to our gross motion planning system in some state x_{k+1}.

We assume that the fine-motion operation can only be performed to pick up a part when the robot has reached the correct source region, and to deliver a part when the robot has reached the

correct destination region. We have considered two alternative ways to map source and destination regions into the configuration space (and state space):

- **Contact:** In the workcell, the robot must only come into contact with the source (or destination) region to apply FMP.
- **Enclosure:** In the workcell, the robot must be completely contained in the source (or destination) region to apply FMP.

When the action FMP is executed, we assume that the assembly mode changes with probability one. At a source region, $m_k = \langle p, s, d, W \rangle$ changes to $m_{k+1} = \langle p, s, d, C \rangle$, and at a destination region, $m_k = \langle p, s, d, C \rangle$ changes to $m_{k+1} = NR$. In Section 6 discusses how the model can be extended for error-handling by defining failure modes in case FMP is not satisfactorily executed.

A *state transition distribution* is defined as $P(x_{k+1}|x_k, u_k)$. This represents a probability distribution over a finite set of next states, given x_k as the initial state, and an action u_k. This relationship is probabilistic because the final component of the state vector (which corresponds to the assembly mode) cannot be completely predicted. Since the remaining components of the state space correspond to the configuration space of the robot, we assume that these can be predicted once x_k and u_k are given. This implies that we have perfect control of the robot (i.e., the response of the robot to a given command is assumed to be executed by an exact, deterministic relationship). In addition, the use of x_k in the conditional of the state transition distribution implies that the robot has perfect information regarding its current state. These choices were made to focus entirely on the most important form of uncertainty for assembly. Section 6, discusses how other forms of uncertainty can be incorporated into our approach.

We present a state transition distribution that applies to the case in which $\mathcal{C} \subseteq \Re^2$, and the robot is limited to translational motion. More complicated motions will be considered in Section 5, including modeling of a redundant manipulator. The motion of the robot could also strongly depend on the assembly mode; for example, the velocity bound, $\|v\|$, might depend on the part that the robot is carrying. We define the action space as $U = [0, 2\pi) \cup \{\emptyset, FMP\}$. If $u_k \in [0, 2\pi)$, then $\mathcal{A}$ attempts to move a distance $\|v\|\Delta t$ toward a direction in $\mathcal{C}$, in which $\|v\|$ denotes some fixed speed for $\mathcal{A}$. If $u_k = \emptyset$, then the robot remains motionless.

Consider the case in which $x_k \in \mathcal{C}_{free}$ is at a distance of at least $\|v\|\Delta t$ from the obstacles. If $\mathcal{A}$

chooses action $u_k \neq \emptyset$ from state x_k, then[1]

$$x_{k+1} = \begin{bmatrix} x_k[1] + \|v\|\Delta t \; cos(u_k) \\ x_k[2] + \|v\|\Delta t \; sin(u_k) \\ m_{k+1} \end{bmatrix}, \tag{5}$$

in which the assembly mode m_{k+1} is known to be sampled from $P(m_{k+1}|x_k, u_k)$. If $u_k = \emptyset$, then $x_k[1] = x_{k+1}[1]$ and $x_k[2] = x_{k+1}[2]$; however, m_{k+1} is not necessarily equal to m_k because the assembly transition equation determines m_{k+1}. We prohibit the robot from considering actions that produce an obstacle collision; however, one could also consider compliant or constrained motions [24, 31, 32, 40].

We now define the notion of a robot strategy for our context. A *strategy at stage* k of $\mathcal{A}$ is a function $\gamma_k : X \rightarrow U$. For each state, x_k, the function γ_k yields an action $u_k = \gamma_k(x_k)$. The set of mappings $\{\gamma_1, \gamma_2, \ldots, \gamma_K\}$ is denoted by γ and termed a *strategy*. This is equivalent to a control law or policy in stochastic control theory [21]. For the examples that we present in this paper, γ_k will be the same for all k (i.e., each robot action depends only on the current state, and not the particular stage). In Section 6 we discuss how assembly situations that require the γ_k to be a function of time may also be handled by extending our model.

3.3 Evaluating robot performance

This section describes how a strategy is evaluated. A *loss functional*,

$$L(x_1, ...x_K, u_1, ..., u_K) = \sum_{k=1}^{K} l_k(x_k, u_k), \tag{6}$$

is defined in a form that is often used in stochastic control theory [21]. Each of the K terms corresponds to costs that are received at a single stage during the execution of the strategy. The ultimate goal of the planner is to determine an optimal strategy $\gamma^* = \{\gamma_1^*, \gamma_2^*, \ldots, \gamma_K^*\}$ that causes L to be minimized in an expected sense[2].

A specific form for l_k is now defined. Let $\bar{t}_f(x_k)$ denote the *expected* time to complete a fine-motion planning task (which results in a new assembly mode) by choosing the action $u_k = FMP$

[1] We use the notation $x_k[i]$ to refer to the i^{th} element of the vector x_k.

[2] The optimal solutions will technically exist in the closure of the free configuration space, as in [23] for a basic motion planning problem; however, we do not consider these topological issues since numerical computation is performed.

from state x_k. Figure 5 illustrates how the exact position where the motion planning "switches" from gross-motion planning to fine-motion planning effects the (expected) time to completing the fine operation. By accounting for this dependence we can improve the overall motion plan, especially when the time for fine-motion planning is significant as is typical in assembly. Recall that x_k simultaneously represents q_k, p, s, and d and C/W. If $m_k \in M_w$ then $\bar{t}_f(x_k)$ represents the expected time to grasp the part. If $m_k \in M_c$ then $\bar{t}_f(x_k)$ represents the expected time for an ungrasp operation for the part (mating with a subassembly, machining, or some other fine motion). In general we have

$$l_k(x_k, u_k) = \begin{cases} 0 & m_k \in NR \\ \bar{t}_f(x_k) & u_k = FMP \\ \Delta t & \text{Otherwise} \end{cases} . \tag{7}$$

The cost that is minimized for our problem thus becomes the aggregate of times that parts wait before being delivered. If there are no requests (i.e., $m_k = NR$), then no penalty is received. To reduce the loss over a long period of time, the robot will prefer actions that bring the assembly mode back to NR as quickly as possible.

This loss functional can also be used to derive appropriate strategies when the source regions are disconnected. In terms of scheduling, having multiple disjoint source regions may be considered as an implementation of a buffer. The interpretation is that there are multiple locations from which a part can be picked up. The robot must make decisions that optimize the loss functional. Decisions of this nature do not affect the stability of the global scheduling algorithm since the corresponding constraints are not affected.

Analogously, our framework can handle multiple disjoint destination regions. The interpretation of this is that to complete a given fine-motion planning task (e.g., mating) there could be disjoint regions in the configuration space that thus form the goal for the gross-motion planning. See, for example, Figure 6 which shows the two directions from which the part can be inserted into the hole in the subassembly thus giving rise to two disjoint destination regions. In Section 5, we show computed examples that contain disjoint sources and destinations.

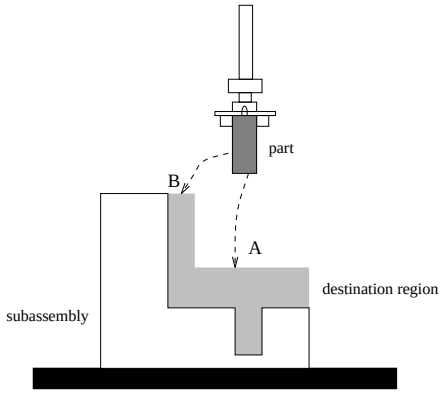


Figure 5: An example of the variation of the cost of the fine motion planning depending on the contact position with the destination region. Contact at A will give rise to a smaller expected time for mating compared to B.

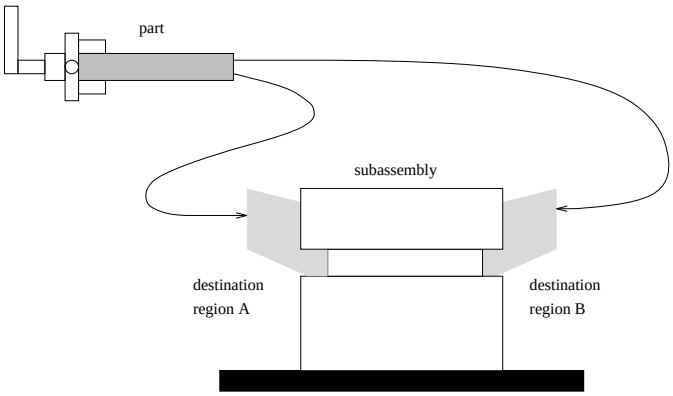


Figure 6: An example where the destination for the gross motion planning is split into two disjoint regions from where the fine motion of mating the part into the subassembly can occur.

4 Computational Scheme

One of the primary advantages of our framework is that a straightforward computation procedure can be used to determine optimal strategies. In this section, we show how the principle of optimality can be applied to our problem to obtain solutions through dynamic programming, and discuss the related computational issues.

4.1 Applying the Principle of Optimality

Suppose that for some k, the optimal strategy is known for each stage $i \in \{k, \ldots, K\}$. The expected loss obtained by starting from stage k, and implementing the portion of the optimal strategy, $\{\gamma_k^*, \ldots, \gamma_K^*\}$, can be represented as

$$\bar{L}_k^*(x_k) = E\left\{\sum_{i=k}^{K} l_i(x_i, \gamma_i^*(x_i))\right\}, \tag{8}$$

in which $E\{\}$ denotes expectation. The expectation is taken over the possible assembly sequences, $\mathbf{m}$. The function $\bar{L}_k^*(x_k)$ is sometimes referred to as the *cost-to-go* function in dynamic optimization literature [3].

The principle of optimality [21] states that $\bar{L}_k^*(x_k)$ can be obtained from $\bar{L}_{k+1}^*(x_{k+1})$ by the following recurrence:

$$\bar{L}_k^*(x_k) = \min_{u_k}\left\{l_k(x_k, u_k) + \sum_{x_{k+1}} \bar{L}_{k+1}^*(x_{k+1}) P(x_{k+1}|x_k, u_k)\right\}. \tag{9}$$

Note that the sum in (9) is taken over a finite number of states, which can be reached using (5).

The goal is to determine the optimal action, u_k, for every value of x_k, and every stage $k \in \{1, \ldots, K\}$. One can begin with stage $K+1$, and repeatedly apply (9) to obtain the optimal actions. At stage $K+1$, we declare that $\bar{L}_{K+1}^*(x_{K+1}) = 0$. The cost-to-go, $\bar{L}_K^*$, can be determined from $\bar{L}_{K+1}^*$ through (9). Using the $u_K \in U$ that minimizes (9) at x_K, we define $\gamma_K^*(x_K) = u_K$. We then apply (9) again, using $\bar{L}_K^*$ to obtain $\bar{L}_{K-1}^*$ and γ_{K-1}^*. These iterations continue until $k = 1$. Finally, we take $\gamma^* = \{\gamma_1^*, \ldots, \gamma_K^*\}$.

The loss function $\bar{L}_1^*(x_1)$ shares similarities with the concept of a global navigation function in motion planning [23, 34]. Also, different forms of dynamic programming have been successfully applied to many other motion planning problems [2, 15, 38]; for instance, the *wavefront expansion method* that is described in [23] can be viewed as a special form of dynamic programming.

4.2 Computational Issues

In our implementation, we determine optimal strategies numerically, by successively building approximate representations of $\bar{L}_k^*$ through the use of (9). This offers flexibility, since analytical solutions are very difficult to obtain for gross motion planning problems with this form of uncertainty, and have only been previously obtained by considering very specific cases [35, 36]. Each dynamic programming iteration can be considered as the construction of an approximate representation of $\bar{L}_k^*$. We decompose the state space into cells of uniform size; however, it is important to note the differences between the use of this decomposition in our context and the traditional use of decompositions in geometric motion planning. Our primary interest in using the decomposition is to construct a good approximation of the function $\bar{L}_k^*$.

We obtain the value for $\bar{L}_k^*(x_k)$ by computing the right side of equation (9) for various values of u_k, including $u_k = \emptyset$. Values of u_k that cause collision are excluded from consideration. To increase computational performance, we compute a binary bitmap representation of the configuration space for each assembly mode. Such representations are fast and straightforward to use [18]. Much of the configuration space is identical for different modes. For example, the static obstacles lead to the same $\mathcal{C}$-space collision regions for NR and all modes in M_w. The value for $\bar{L}_k^*(x_k)$ is obtained by linear interpolation between neighboring cells. This significantly reduces the level of resolution that must be considered. Other schemes, such as quadratic interpolation, can be used to further improve numerical accuracy [22].

After some finite number of iterations, for every state, the optimal action becomes fixed with respect to additional iterations. The resulting optimal strategy is considered *stationary*, since it only depends on the state, as opposed to additionally requiring the stage index. Note that no choice of K is necessary as long it is larger than the number of iterations involved in the convergence. Also, at each iteration of the dynamic programming algorithm, we only retain the representation of $\bar{L}_{k+1}^*$ while constructing $\bar{L}_k^*$.

To execute a strategy, final cost-to-go representation (called $\bar{L}_1^*$) is used. The robot is not confined to move along the quantization grid that is used for determining the cost-to-go functions. The optimal action can be obtained from any real-valued location $x \in X$ though the use of (9), linear interpolation, and the approximate representation of $\bar{L}_1^*$. A real-valued initial state is given (the final component represents the assembly mode, and is an integer). The application of the

optimal action will yield a new real-valued configuration for the robot.

To evaluate computational performance, there are two phases to consider: determination of the optimal strategy, and execution of the optimal strategy. The iterated dynamic programming computations are performed off-line to determine the optimal strategy. The time complexity is linear in both the number of quantized points and the assembly modes; however, it is exponential in the dimension of the state configuration space, as is typically the case for approaches to basic motion planning problems that do not involve uncertainty (see [23]). For problems in two-dimensional configuration space, this off-line computation takes a couple of minutes, while three-dimensional configuration-space problems take a few hours. These times correspond to a prototypical implementation of the planner on a standard SPARC 10 workstation. We typically use about 50 cells per dimension of the configuration space. Significant improvement of these off-line computations can be obtained through additional code optimization and parallelization; however, these implementation issues are beyond the focus of this research.

The on-line execution of the optimal strategy proceeds very quickly. For each stage, a single evaluation of the dynamic programming equation is performed to yield the optimal action. This computation is on the order of a few milliseconds, and is therefore quite reasonable for practical applications.

5 Solutions for Specific Assembly Situations

In this section we present computed solutions for six different problems that involve the transfer of parts in a workcell for assembly. The problems are chosen to illustrate the flexibility and generality of our approach. The first four problems involve a rigid robot in the workcell. The rigid robot could be the representation of the end-effector that is relevant for gross-motion planning. Examples of industrial robot systems for which the results could be applied (by considering only the relevant joints) are the SCARA-type robots, e.g., AdeptOne of the Adept Technology Inc. or a Cartesian robot, e.g., the IBM 7565 robotic system. The final two problems involve three-link articulated-manipulators, for which optimal strategies are derived directly on the joint space. The three-dimensional configuration space used in the examples has the same complexity as the first three joints of a PUMA-type robot or a SCARA-type robot [30]. This section concludes with a discussion

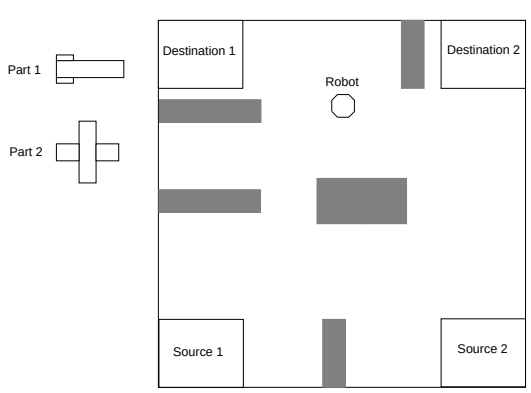
of the benefits that were observed in our simulations.

5.1 Rigid-robot simulations

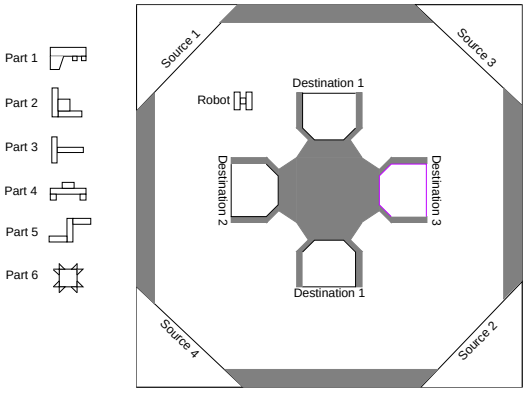
Problem 1. The first example is designed to illustrate many of the basic concepts. It involves a rigid robot that translates in a planar workcell cluttered with obstacles (see Figure 7). There are two different parts that can be moved from either of two sources to either of two destinations. There are consequently 17 possible assembly modes. The probability that a request will appear at stage $k+1$ while $m_k = NR$ is given to be 0.05. All the p, s, d combinations are equally likely to occur. We assume that once a p, s, d combination is given to the robot, it will not change or be retracted until part p is delivered to destination d. The robot moves with $\|v\|\Delta t = 3.0$ under the motion equation (5), with workcell being 100 units square.

Figures 8.a and 8.b depict the level-set contours of the cost-to-go function, $L_1^*(x_1)$ for assembly modes $\langle 1, 1, 1, W\rangle$ and $\langle 1, 1, 1, C\rangle$, respectively. In Figure 8.a there is a minimum at the first source region, and in Figure 8.b a minimum appears at the destination region. For translational motion, the negative gradient of the cost-to-go function represents the direction of motion of the robot. Hence, the cost-to-go function is similar to a numerical navigation function [19, 23, 34]. Figures 8.c and 8.d depict the optimal strategy γ^* for assembly modes $\langle 1, 1, 1, W\rangle$ and $\langle 1, 1, 1, C\rangle$, respectively. The direction of each arrow indicates the direction of motion (specified as $u_k = \gamma^*(x_k)$) for the robot, from that particular state. The motion directions are shown at fewer state locations than appear in the machine implementation to add clarity to the figure. The places in which there are no arrows correspond to configurations in which the robot (or possibly the part) is in collision with a static obstacle.

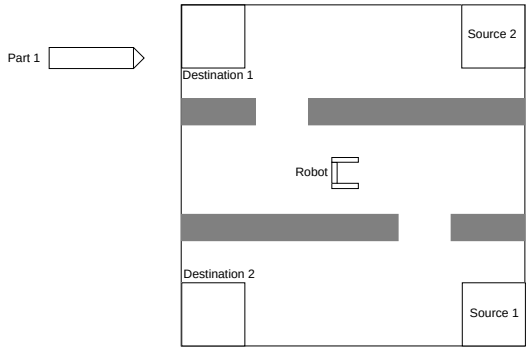
Figure 9 presents a simulation of the robot in the workcell over a period of time, under the implementation of γ^*. The beginning of the trajectory is depicted in Figure 9.a, and it concludes in Figure 9.i. To save space in the figure, many frames are superimposed, and a new picture is shown only when the assembly mode changes. The first column of Figure 9 corresponds to execution during the NR mode. The second column corresponds to modes in M_w, and the final column corresponds to modes in M_c. In the last two columns, the source and destination regions that correspond to the issued request are shaded. In the final column, the part that is carried by the robot is shaded in black. There are at least two interesting behaviors to note in this solution.



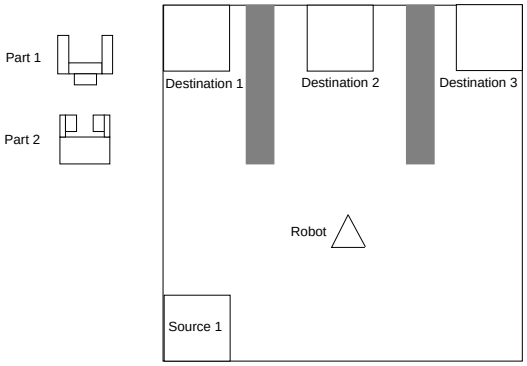
Problem 1



Problem 2



Problem 3



Problem 4

Figure 7: Four problems involving a rigid robot, several parts, source regions, and destination regions. The details of the model used for gross-motion planning is given in the text.

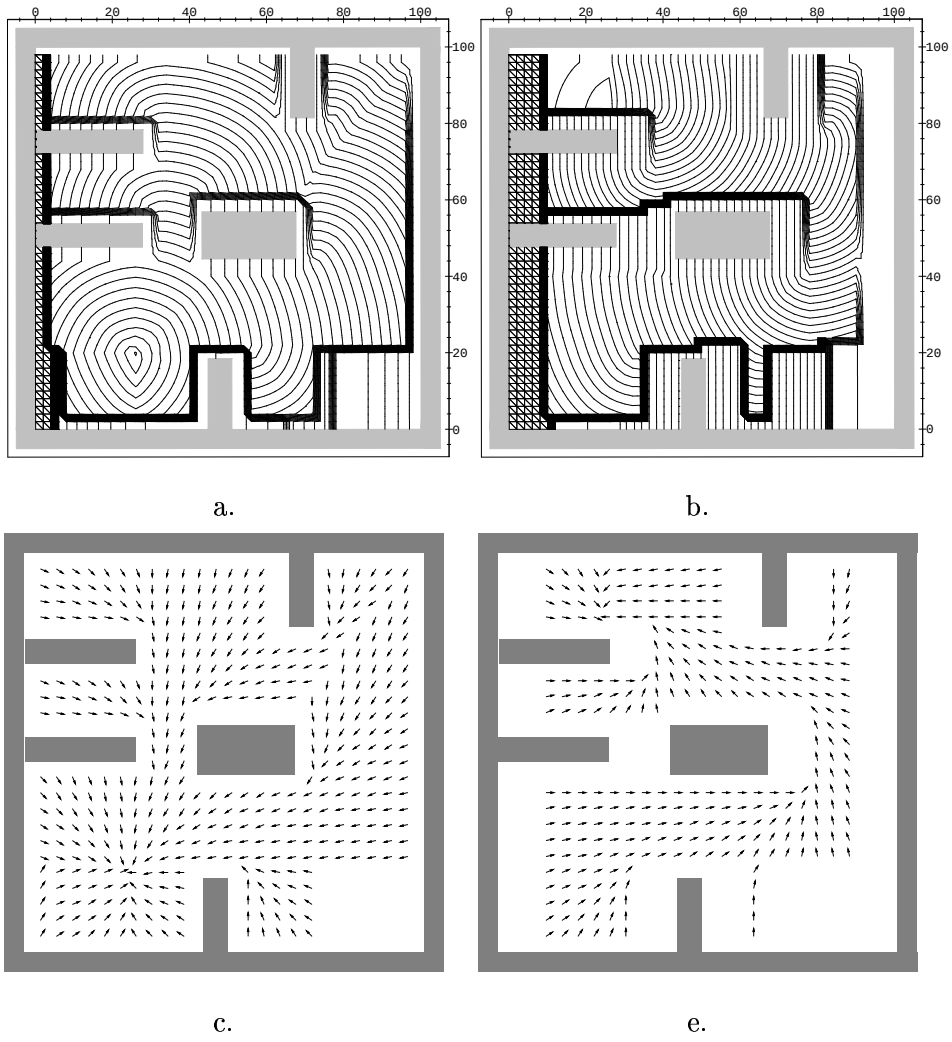


Figure 8: a) Level-set contours of the cost-to-go function for $e = \langle 1, 1, 1, W \rangle$; b) the contours for $e = \langle 1, 1, 1, C \rangle$; c) the optimal actions as a vector field for $e = \langle 1, 1, 1, W \rangle$; d) the optimal actions for $e = \langle 1, 1, 1, C \rangle$

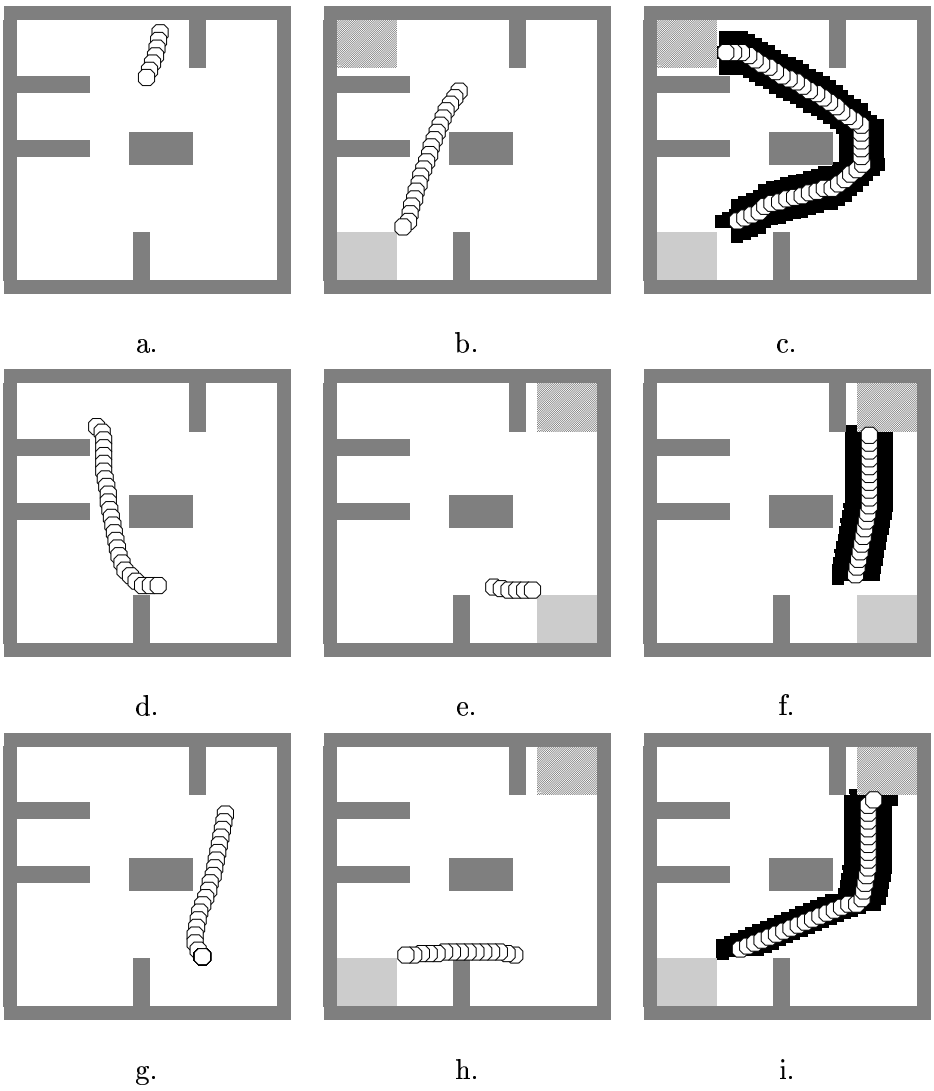


Figure 9: A simulation result under the implementation of the optimal strategy γ^* for Problem 1.

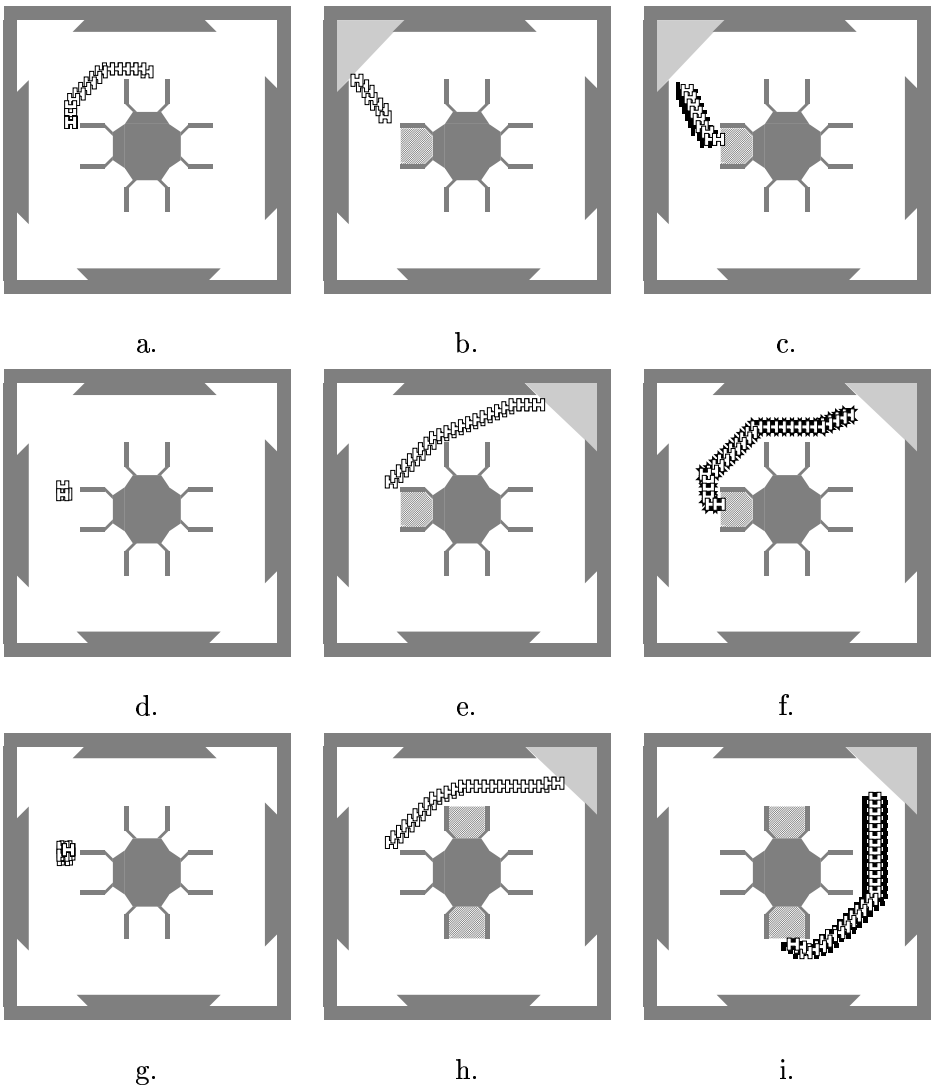


Figure 10: A simulation result under the implementation of the optimal strategy γ^* for Problem 2.

When the assembly mode is NR, the robot moves to a location in the lower portion of the workcell. This behavior naturally occurs through the optimization of the criterion. It is best for the robot to wait near sources while there are no requests, to reduce the expected time to deliver a part that might appear. This corresponds to reducing the setup time in a scheduling system, and is hence a preferred behavior for the robot. Another behavior to note is how the changing geometry affects the trajectory of the robot. In Figures 9.b and 9.d the robot does not carry a part, and hence is able to move through a narrow opening. However, in Figure 9.c the robot carries a part, and consequently must take a longer route to reach the destination.

For the remaining problems in this section, we will show figures that indicate the sample path under the implementation of the optimal strategy, and are similar to Figure 9.

Problem 2. This example (Figure 7) involves a translating robot in which there are 6 parts, 4 sources, and 3 destinations. In addition, Destination 1 has two disconnected components, and the robot must choose the best delivery point to reduce loss. For this problem there are 72 different kinds of requests (which are equally likely to occur), which results in 145 assembly modes. Figure 10 shows a sample of the execution under γ^*. Note the behavior of the robot with respect to the disconnected components of the Destination 1. At the start of the time period captured in Figure 10.h, the robot receives a request to move Part 6 from Source 3 to Destination 1. The robot picks up the part from Source 3, and chooses to deliver it to the lower component of the Destination 1 (Figure 10.i). This behavior was based on the computation of the optimal strategy for that particular position of the robot in the NR mode.

Problem 3. In the previous two examples, the robot was only allowed to translate. This helped in illustrating the solution in greater detail for Problem 1 and in allowing us to study a more complex assembly situation involving fairly large number of assembly modes in Problem 2. In Problem 3 we additionally allow the robot to rotate, thus making the state space three-dimensional. This example (Figure 7) involves an assembly situation in which there is a rotating robot, 1 part, 2 sources, and 2 destinations. We must update equation (5) to model the rotational motion of the robot. We assume that the robot can rotate in place, or translate along its axis of orientation. Other motion models (such as nonholonomic, radius-constrained motion) that are compatible with the framework presented in this work can be found in [25]. Figure 11 shows a sample of the execution

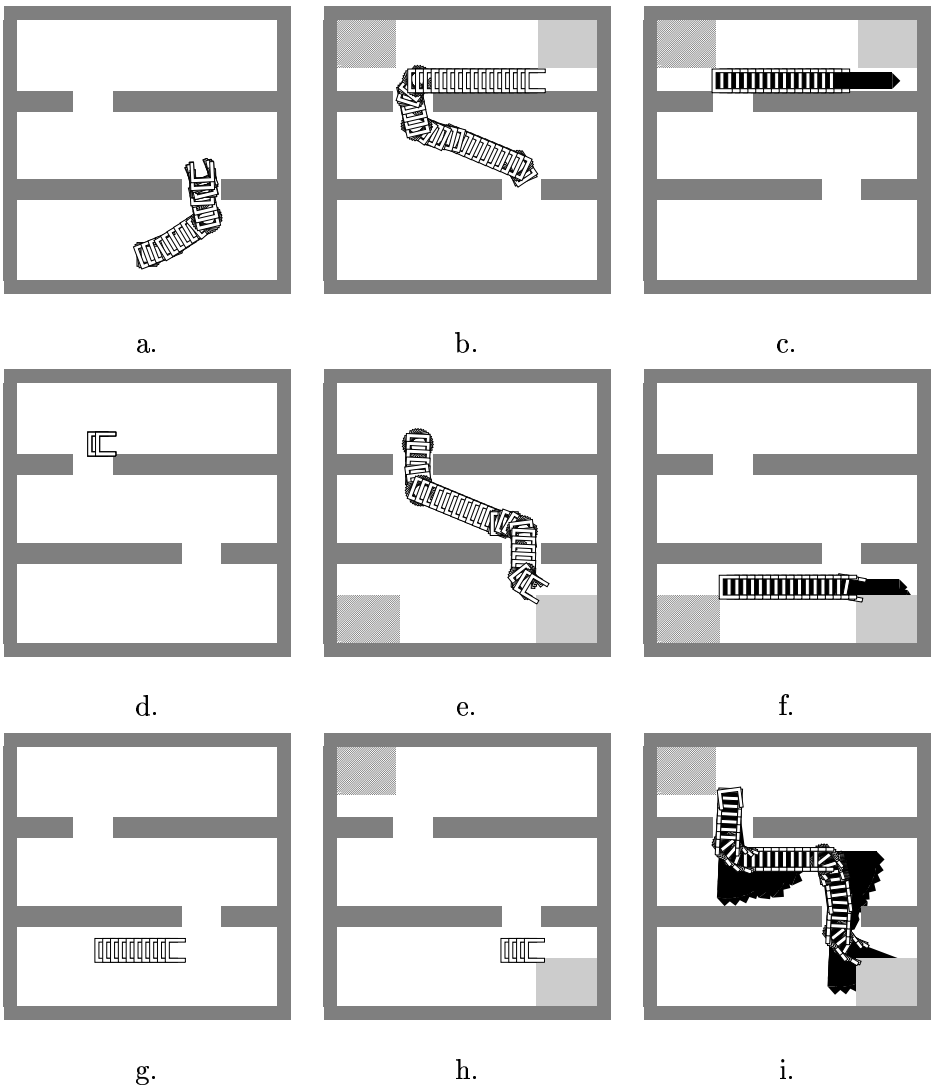


Figure 11: A simulation result under the implementation of the optimal strategy γ^* for Problem 3.

under γ^*. Because the manner in which the obstacles are arranged and because the part that the robot could carry is large relative to the opening, the optimal position of the robot in the NR mode (Figure 11.a) is important since it can significantly affect the carrying time when the request arrives. In this problem there are 4 possible request (p, s, d combinations) and 9 assembly modes.

Problem 4. In all the three problems discussed so far, the stochastic model of the assembly process was defined in terms of the transition probabilities from the NR mode of the robot. In Problem 4 we discuss an alternative stochastic model that defines the transition probabilities with respect to the destination regions (as discussed in Section 3.2). The specific feature that this model induces on the behavior of the robot is its ability to change destinations *while* it is already carrying a part. Problem 4 (Figure 7) considers an assembly situation involving a translating robot with 2 parts, 1 source, and 3 destinations. For this problem, the destination is allowed to change while the robot is carrying a part. The probability that the destination will change in a given stage is 0.02. The probability for changing to each of the other two destinations is 0.01. The robot models the changing destination probabilistically as discussed in Section 3.2. Figure 12 shows a sample of the execution under γ^*. The destination changes during execution depicted in Figure 12.c and in Figure 12.g. In both cases, the robot immediately responds by delivering the part to the new destination. This kind of assembly situation can arise when there is on-line monitoring of the assembly process and the robot has access to the current demand at a particular destination at any given time. For example, suppose the same part is needed by two destination regions $D1$ and $D2$ at a given time. The scheduler schedules the part to be delivered to $D1$, but due to an error there is a delay in its previous operation and $D1$ is not ready for that part while the robot is in the process of carrying it. This would be detected by on-line plan monitoring and the part rescheduled to arrive at $D2$. The probabilistic modeling of such an assembly situation helps in improving the gross-motion planning.

5.2 Articulated-manipulator simulations

In this section we consider motion planning for three-link articulated manipulators performing assembly operations. Several additional concerns must be addressed that pertain to collision detection of the whole arm and the state transition distribution. To define the source and destination regions in the configuration space, we only consider the end-effector of the robot. This is a reasonable

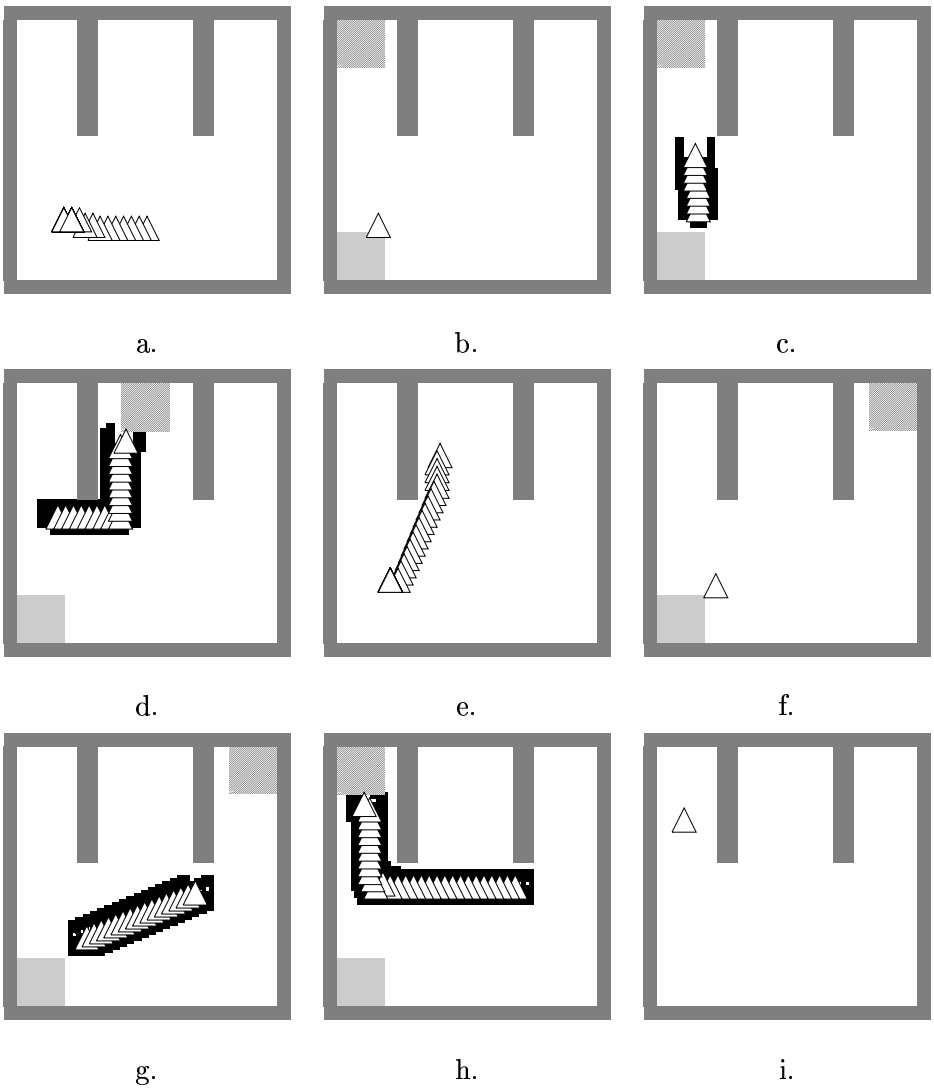


Figure 12: A simulation result under the implementation of the optimal strategy γ^* for Problem 4.

choice since fine-motion planning essentially would involve only the end-effector along with the part that it could be carrying (See Section 3.1).

We define the motion strategies directly in terms of the joints (that can be independently controlled), instead of considering their representation in the workspace. The collision detection for the articulated arm in our implementation is done in terms of the coordinate space of the workcell. For the examples that we consider, the planar robot manipulator has redundant degrees of freedom. This is due to the fact that even though the robot has three degrees of freedom, the source and destination regions lie in the plane.

Problem 5. For the first manipulator problem, there are two parts, two sources, and two destinations (see Problem 5 in Figure 13). There are three links that move in the plane, and an end-effector that always maintains the same orientation. The figure can be considered as a side view of a problem in which objects are to be moved from trays that exist at different levels. There are joint limits that prevent the joints from executing a circular motion. Figure 14 shows a sample of the execution. The third column shows the part being "carried" to the destination region, and the transition to fine-motion planning occurs when the end-effector contacts the destination region.

Problem 6. For the second manipulator problem, there are one part, two sources, and four destinations (see Problem 6 in Figure 13). One of the sources has two disconnected components. There are three links that move in the plane. The problem can be considered as a top view of a workspace in which objects are to be moved between locations on a planar surface. There are fixed limits for each joint. Figure 15 shows a sample of the execution. Note the behavior of the robot induced by the fact that there are two disconnected components for the Source 1. Thus every time a request arrives involving Source 1, the motion of the robot varies depending on its current position and the destination region which is also part of the request. For example, in the segment of its execution captured in Figure 15, there were two requests involving Source 1 (seen in terms of the shaded regions of Figure 15.e and h). For the first such request (second row), the robot chooses to pick the part from the upper component of Source 1, although it was nearer to the lower component of Source 1 when the request arrived. This was due to the fact that the corresponding destination was nearer to the upper component. In the other two cases involving Source 1 (see the third and fourth rows of Figure 15), the robot chose the lower component instead.

5.3 Concluding Remarks on Simulation Studies

In the simulation experiments we repeatedly observed behaviors that indicate the improvement in expected performance over planning for each request as it arrives. The assembly process allows partial prediction of future requests to occur. The strategies obtained by our computational approach optimally incorporate this predictive information. Most of the savings due to partial prediction occur when $m_k = NR$. During this mode, the robot moves to certain locations in the workcell in anticipation of future requests. In a particular instance, the robot might make a poor prediction about where the next source will be; however, on average the robot is guaranteed to exhibit the time-optimal behavior.

Several of the computed examples clearly illustrate the benefits due to prediction. Parts a, d, and g of Figure 9.a show motions that occur while $m_k = NR$. In each case the robot was much closer to the future requested source than it would have been without this anticipatory motion. In comparison with the case when such anticipatory motion is absent, the improvement in performance (measured in stages) is nearly equal to the number of stages that occurred while $m_k = NR$. One might alternatively consider defining heuristics for the behavior of the robot; however, our strategies are selected automatically and guaranteed to yield the best performance in the expected sense. Several other computed examples show similar behaviors; see Figures 11.a and 14.d. While improving the expected performance of the individual robot cell, these optimal motions can be an important factor in maintaining the stability of the entire manufacturing plant (as discussed in Section 2.4).

6 Extension to Other Assembly Situations

Apart from the specific assembly situations presented, our framework is capable of representing a larger class of motion planning problems, making it applicable to more assembly situations. In this section we discuss some of these applications and the modifications to the model that would be necessary to the handle these extensions.

Time-varying models of the assembly process. The motion strategies that have been considered in Section 5 are stationary in the sense that the optimal robot actions depend only on

the state. The optimal strategy for the robot does not depend on time since the model components such as the state transition distribution, or the environment transition probabilities, do not depend on the particular stage index, $k \in \{1, \ldots, K\}$. However, the model components can be allowed to vary over time. This affords the opportunity to model many interesting problems. For example, by allowing the assembly mode transition probabilities to vary, many more statistical processes can be modeled. For instance, if the demand arrival varies according to the stage of some other process in the assembly plant, this can be factored into the motion optimization to improve the performance of the assembly cell. Another possibility is to let the source or destination region move over time as would be true if the assembly part is actually on a conveyor belt (Figure 16) that moves with a constant known velocity. In this case, the robot must "intercept" the moving source or destination region as a terminating condition for the gross-motion planning. In fact, in such a situation, the consideration of fine-motion planning (see Section 3.3) may be even more important than the case with stationary source/destination regions. The tradeoff, however, is that since the optimal strategy depends on stages, more storage is required. There would be a state-space mapping for each stage, which is at least reasonable for two-dimensional configuration space problems, given the current implementation. The space requirements could be significantly reduced if the initial state is known, because at certain stages many portions of the state space will never be reached.

Processing simultaneous requests. As stated in Section 3, we decided to consider only the arrival of a single request at a time, with the assumption that a global scheduler that was external to the workcell provided the requests. We can alternatively process simultaneous requests, and place the scheduling burden on the robot. This requires defining additional assembly modes that correspond to various combinations of requests. The loss functional for this extension can be defined in a number of ways. One reasonable way is to assign one unit of loss for each waiting part that has not yet been delivered during the current stage. This will force the robot to clear the requests as quickly as possible, while making decisions about the order of delivery partly on the basis of the geometry of the motion planning problem. Under this condition, the computational approach remains the same, while the robot additionally performs the scheduling. One implication of this choice, however, is that the most appropriate ordering of part deliveries for the robot might not necessarily correspond to the most appropriate ordering for the manufacturing system. For example,

certain parts might be more urgent than others due to other modules in the manufacturing process, which can only be inferred by a more global analysis.

Assembly with failure modes. Another interesting extension of the model would be for a situation in which *failures* could occur in the assembly operation. Such failures could arise, for example, in machining a part when a preset specification is not met. A subassembly that was supposed to be built might fail because a part breaks or has unacceptable dimensions. Failures of this type can be represented in terms of additional assembly modes and incorporated into our motion planning scheme. The robot would then respond optimally to the failures, under an appropriate probabilistic model of failures.

Incorporating other forms of uncertainty. In this paper we have focused primarily on the form of uncertainty in assembly involving time since this can play a key role in the efficiency of gross motions. The positional and control uncertainty associated with the robot are more relevant in the final stages of the assembly, i.e., for fine-motion planning. However, other forms of uncertainty can be factored into the same probabilistic approach. In fact, the framework presented here can facilitate such a combination. As a first step toward this combination, we incorporated the expected costs involved in the fine-motion planning (see Section 3.3) for evaluating the performance of gross motions. However, the uncertainty involved in fine-motion planning can be included directly into the model. A treatment of additional forms of uncertainty in fine-motion planning (position and control uncertainty) that is compatible with our treatment of the uncertainty with respect to time is reported in [24].

7 Conclusions

Robot motion optimization over time is important since efficient motion planning eventually translates into an increased throughput in an assembly system. Moreover, optimizing the efficiency of the individual robot cell can play a significant role in ensuring the stability of the entire assembly plant. While gross-motion planning is traditionally not considered part of assembly planning, the results in this paper show how the assembly performance can be improved by considering gross-motion planning as part of the assembly process. At one level, this helps in developing an interface

between gross-motion planning and scheduling, and at another level it develops better interface between gross-motion planning and fine-motion planning in the presence of uncertainty. This work has exploited these interactions by delivering a framework for gross-motion planning in an assembly system.

Analytical solutions to stochastic optimal control problems are quite difficult to obtain; thus the dynamic programming-based numerical solution offers a practical alternative. Within the computational limits of the approach, many realistic assembly situations can be treated. This was demonstrated by our simulation studies on a variety of specific assembly situations.

There are several directions for future research, some of which were discussed in the paper. To handle assembly planning problems that have a greater deal of complexity, additional computational techniques may need to be developed. The use of the stochastic assembly process provides a flexible way to capture the time-varying element of assembly operation. However, further research is needed to understand the relationship of this model to other models of uncertainty and scheduling. As illustrated by the specific examples, many different modeling choices are possible and the right choice would depend on the particular assembly situation.

Acknowledgement. We would like to thank P. R. Kumar for helpful discussions on scheduling and to M. Barbehenn for comments on a draft of this paper. We would also like to thank the anonymous reviewers whose insightful comments were a great help in improving the presentation of the paper.

References

[1] R. Alami, T. Siméon, and J. P. Laumond. A geometric approach to planning manipulation tasks. In *Proc. Fifth International Symp. Robotics Research*, pages 113–119, 1989.

[2] J. Barraquand and P. Ferbach. A penalty function method for constrained motion planning. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1235–1242, 1994.

[3] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[4] B. Mc Carragher and H. Asada. A discrete event approach to the control of robotic assembly tasks. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 331–336, 1993.

[5] S.-C. Chang and D.-T. Liao. Scheduling flexible flow shops with no setup effects. *IEEE Transactions on Robotics and Automation*, 10(2):99–111, 1994.

[6] L. M. M. Custodio, J. J. S. Sentieiro, and C. F. G. Bispo. Production planning and scheduling using a fuzzy decision system. *IEEE Transactions on Robotics and Automation*, 10(2):160–168, 1994.

[7] T. L. DeFazio and D. E. Whitney. Simplified generation of all mechanical assembly sequences. *IEEE Journal of Robotics and Automation*, 3(6):640–658, 1987.

[8] B. R. Donald. *Error Detection and Recovery in Robotics.* Springer-Verlag, 1989.

[9] M. Erdmann. Using backprojections for fine motion planning with uncertainty. *International Journal of Robotics Research*, 5(1):19–45, 1986.

[10] A. Fox and S. Hutchinson. Exploiting visual constraints in the synthesis of uncertainty-tolerant motion plans. *IEEE Transactions on Robotics and Automation*, 11:56–71, 1995.

[11] S. Gottschlich, C. Ramos, and D. Lyons. Assembly and task planning: A taxonomy. *IEEE Robotics and Automation Magazine*, 1(3):4–12, 1994.

[12] J. A. Hendler and J. C. Sanborn. Planning and reaction in dynamic domains. In *Proc. DARPA Workshop on Knowledge-Based Planning Systems*, 1987.

[13] L. S. Homem de Mello and Sukhan Lee. *Computer-Aided Mechanical Assembly Planning.* Kluwer Academic, 1991.

[14] L. S. Homem de Mello and A. C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Journal of Robotics and Automation*, 7(2):228–240, 1991.

[15] H. Hu, M. Brady, and P. Probert. Coping with uncertainty in control and planning for a mobile robot. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1025–1030, 1991.

[16] Y. F. Huang and C. S. G. Lee. A framework of knowledge-based assembly planning. In *Proc. IEEE International Conf. on Robotics and Automation*, pages 599–604, 1991.

[17] H. Inuoe. *Force Feedback in Precise Assembly Tasks*, volume 2, pages 219–241. MIT Press, 1981.

[18] L. E. Kavraki. Computation of configuration-space obstacles using the fast fourier transform. *IEEE Transactions on Robotics and Automation*, 11:408–412, 1995.

[19] D. Koditschek. Robot planning and control via potential functions. In O. Khatib, J. J. Craig, and T. Lozano-Pérez, editors, *The Robotics Review 1.* MIT Press, 1989.

[20] V. S. Kouikoglou and Y. A. Phillis. Discrete event modeling and optimization of unreliable production lines with random rates. *IEEE Transactions on Robotics and Automation*, 10(2):153–159, 1994.

[21] P. R. Kumar and P. Varaiya. *Stochastic Systems.* Prentice Hall, Englewood Cliffs, NJ, 1986.

[22] R. E. Larson and J. L. Casti. *Principles of Dynamic Programming, Part II.* Dekker, New York, NY, 1982.

[23] J. C. Latombe. *Robot Motion Planning.* Kluwer Academic, Boston, MA, 1991.

[24] S. M. LaValle and S. A. Hutchinson. An objective-based stochastic framework for manipulation planning. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1994.

[25] S. M. LaValle and R. Sharma. A framework for motion planning in stochastic environments: Applications and computational issues. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3063–3068, May 1995.

[26] D. Y. Lee and F. DiCesare. Scheduling flexible manufacturing systems using Petri nets and heuristic search. *IEEE Transactions on Robotics and Automation*, 10(2):123–132, 1994.

[27] S. Lee. Backward assembly planning with assembly cost analysis. In *Proc. IEEE International Conf. on Robotics and Automation*, pages 2382–2391, 1992.

[28] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 4:3–24, 1985.

[29] S. H. Lu and P. R. Kumar. Distributed scheduling based on due dates and buffer priorities. *IEEE Trans. on Automatic Control*, 36(12):1406–1416, 1991.

[30] H. Makino and N. Furuya. Scara robot and its family. In *Proc. International Conference on Assembly Automation*, pages 433–444, 1982.

[31] M.T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Trans. on Systems, Man, and Cybernetics*, 11(6):418–432, 1981.

[32] R. P. Paul and B. Shimano. Compliance and control. In *Proc. of the Joint American Automatic Control Conference*, pages 1694–1699, 1976.

[33] J. R. Perkins, C. Humes Jr., and P. R. Kumar. Distributed scheduling of flexible manufracturing systems: Stability and performance. *IEEE Transactions on Robotics and Automation*, 10(2):133–141, 1994.

[34] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, October 1992.

[35] R. Sharma. Locally efficient path planning in an uncertain, dynamic environment using a probabilistic model. *IEEE Transactions on Robotics and Automation*, 8(1):105–110, February 1992.

[36] R. Sharma, D. M. Mount, and Y. Aloimonos. Probabilistic analysis of some navigation strategies in a dynamic environment. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(5):1465–1474, September 1993.

[37] M. Spong and I. Vidyasagar. *Robot Dynamics and Control.* John Wiley & Sons, 1989.

[38] S.-H. Suh and K. G. Shin. A variational dynamic programming approach to robot-path planning with a distance-safety criterion. *IEEE Transactions on Robotics and Automation*, 4(3):334–349, 1988.

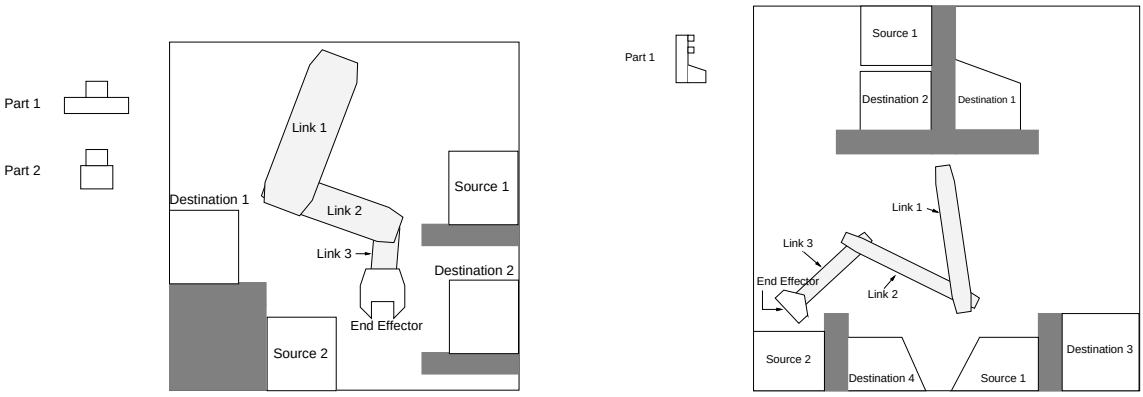
[39] C. van Delft. Approximate solutions for large-scale piecewise deterministic control systems arising in manufacturing flow control models. *IEEE Transactions on Robotics and Automation*, 10(2):142–152, 1994.

[40] D. E. Whitney. Force feedback control of manipulator fine motions. *Journal of Dynamic Systems, Measurement, and Control*, 98:91–97, 1977.

[41] G. Wilfong. Motion planning in the presence of movable obstacles. In *Proc. ACM Symposium on Computational Geometry*, pages 279–288, June 1988.

[42] R. H. Wilson. *On Geometric Assembly Planning.* PhD thesis, Stanford University, March 1992.

[43] Q. Zhu. Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 7(3):390–397, 1991.



Problem 5 Problem 6

Figure 13: Two assembly planning problems involving articulated manipulators with three links. The details of the model used for gross-motion planning is given in the text.

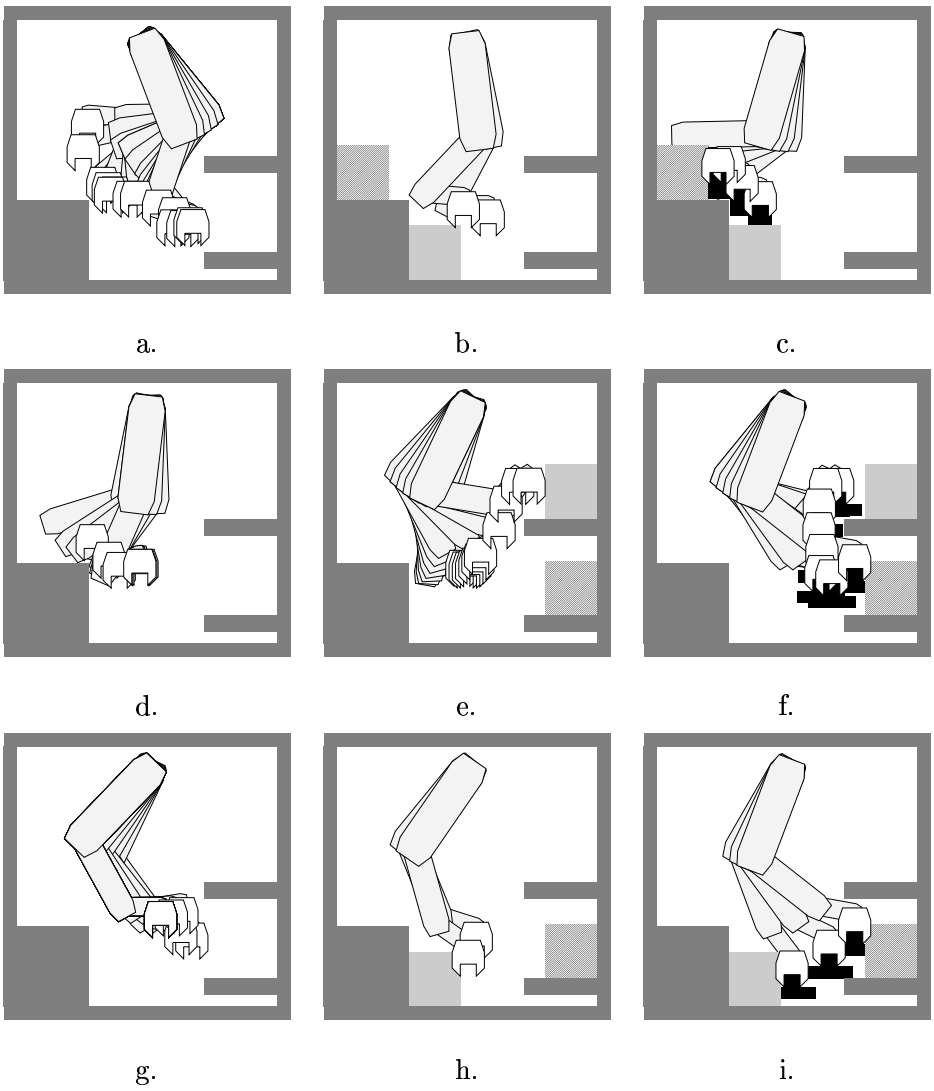


Figure 14: A simulation result under the implementation of the optimal strategy γ^* for Problem 5.

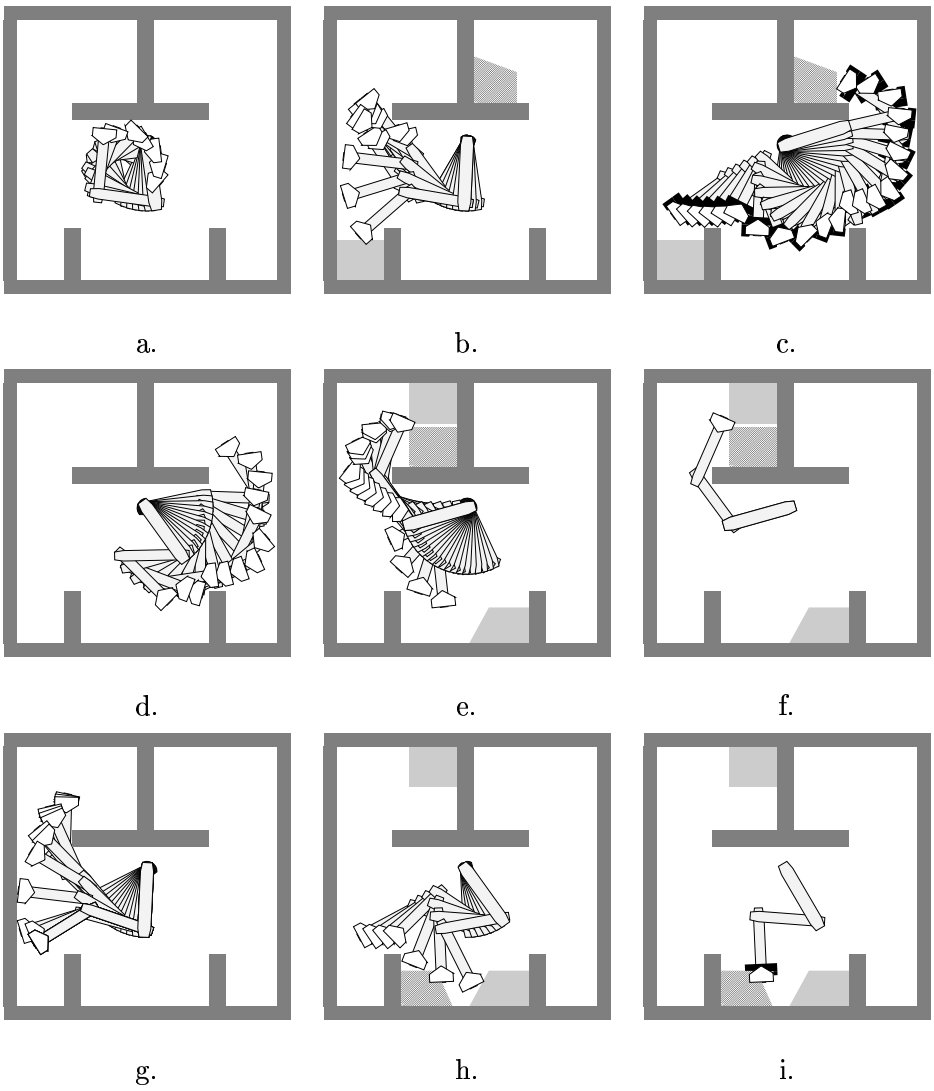


Figure 15: A simulation result under the implementation of the optimal strategy γ^* for Problem 6.

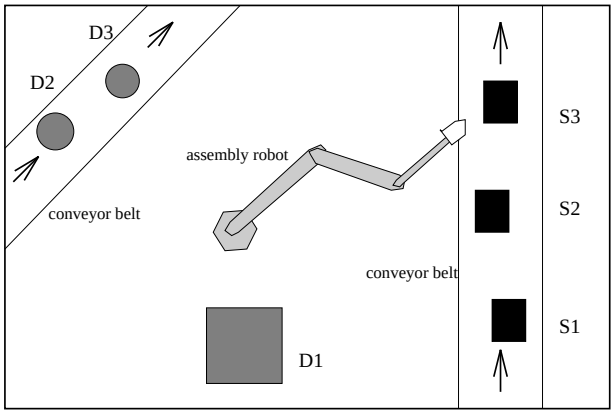


Figure 16: Motion planning for assembly when parts are picked up from and mated to subassemblies on conveyor belts.

List of Figures

1 An assembly plant with multiple robot cells. P's are the parts, B's are the subassemblies, A's are the assemblies and RC's are the assembly robot workcells. . . . . .
2 The motion planning problem in the robot workcell RC1, for multiple assembly from multiple components, with the robot R1 and the additional obstacles O1, O2, ans O3..
3 An example of (a) a source region and (b) a destination region, used for defining the gross-motion planning problem, while establishing good preconditions for fine-motion planning. . . . . .
4 The abstract representation of the motion planning with the assembly in mode $\langle p, s, d, C/W \rangle$, the last component being C when the robot is carrying the part and W otherwise. . . . . .
5 An example of the variation of the cost of the fine motion planning depending on the contact position with the destination region. Contact at A will give rise to a smaller expected time for mating compared to B. . . . . .
6 An example where the destination for the gross motion planning is split into two disjoint regions from where the fine motion of mating the part into the subassembly can occur. . . . . .
7 Four problems involving a rigid robot, several parts, source regions, and destination regions. The details of the model used for gross-motion planning is given in the text.
8 a) Level-set contours of the cost-to-go function for $e = \langle 1, 1, 1, W \rangle$; b) the contours for $e = \langle 1, 1, 1, C \rangle$; c) the optimal actions as a vector field for $e = \langle 1, 1, 1, W \rangle$; d) the optimal actions for $e = \langle 1, 1, 1, C \rangle$ . . . . . .
9 A simulation result under the implementation of the optimal strategy γ^* for Problem 1. . . . . .
10 A simulation result under the implementation of the optimal strategy γ^* for Problem 2. . . . . .
11 A simulation result under the implementation of the optimal strategy γ^* for Problem 3. . . . . .
12 A simulation result under the implementation of the optimal strategy γ^* for Problem 4. . . . . .
13 Two assembly planning problems involving articulated manipulators with three links. The details of the model used for gross-motion planning is given in the text. . . . .
14 A simulation result under the implementation of the optimal strategy γ^* for Problem 5. . . . . .
15 A simulation result under the implementation of the optimal strategy γ^* for Problem 6. . . . . .
16 Motion planning for assembly when parts are picked up from and mated to subassemblies on conveyor belts. . . . . .