Saitou, K., Jakiela, M. J., "Subassembly Generation via Mechanical Conformational Switches," *Artificial Life* **2**(4): 377–416, 1995.

Subassembly Generation via Mechanical Conformational Switches

Kazuhiro Saitou* **Mark J. Jakiela**
Computer-Aided Design Laboratory
Massachusetts Institute of Technology

Abstract

A question is posed on how a particular subassembly sequence is generated in randomized assembly. An extended design of mechanical conformational switches [16] is proposed, which can encode several subassembly sequences. A particular subassembly sequence is generated due to conformational changes of parts during one-dimensional randomized assembly. The optimal subassembly sequence that maximizes the yield of a desired assembly can be found via genetic search over a space of parameterized conformational switch designs, rather than a space of subassembly sequences. The resulting switch design encodes the optimal subassembly sequence so that the desired assemblies are put together only in the optimal sequence. The results of genetic search and rate equation analyses reveal that the optimal subassembly sequence depends on the initial concentration of parts and the defect probabilities during randomized assembly. The results indicate that abundant parts and parts with high defect probabilities should be assembled earlier rather than later.

Keywords: subassembly generation, mechanical conformational switches, randomized assembly, genetic optimization.

1 Introduction

1.1 Conformational switching in bacteriophage assembly

Bacteriophages are a type of virus which infect bacterial cells. Upon infection, they take over a cell's replication mechanism to synthesize their component molecules such as nucleic acids and proteins. Spontaneous self-assembly of the component proteins then occurs which produces hundreds of new progeny viruses in the host cell. New viruses are formed via assembly of randomly arranged and randomly moving component protein molecules. In the assembly process, however, the various component molecules do not associate with one another at random. Instead, the assembly occurs in a definite sequence, or in a fixed *morphogenetic pattern*. Biologists believe that *conformational switches* in protein molecules facilitate this randomized assembly of bacteriophages. In a protein molecule with several bond sites, a conformational switch causes the formation of a bond at one site to change the conformation of another bond site. As a result, a conformational change which occurred at an assembly step provides the essential substrate for assembly at the next step [19].

This process of randomized assembly may have practical applications in mechanical engineering, such as part orientation for high-volume assembly, and assembly of micro electromechanical systems (MEMS). As in the case of biological randomized assembly, conformational switches could also play an important role in such randomized mechanical assembly. Our previous work on evolutionary design of mechanical conformational switches [16] was our first attempt toward understanding randomized conformational assembly of mechanical parts. In this paper, we will extend that work to describe subassembly generation in randomized mechanical assembly.

*Author to whom correspondence should be addressed: Massachusetts Institute of Technology, Department of Mechanical Engineering, 77 Massachusetts Avenue, Room 3-452, Cambridge, MA 02139; Phone: (617) 258-8186; email: kazu@mit.edu.

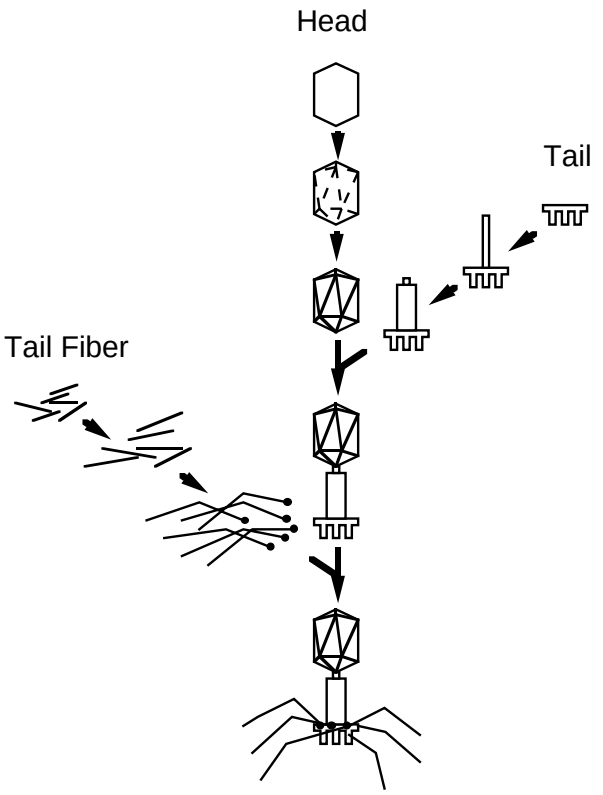


Figure 1: the pathway of T4 bacteriophage assembly (simplified and rearranged from Figure 3 of [17])

1.2 Principle of subassembly

Assembly of many complex systems – either biological or mechanical – takes place in several stages. At each assembly stage, *subassemblies* are made, which are then incorporated into subassemblies at the next stage. For example, the beginning stage of an automobile assembly is the production of elementary parts, such as wire, bolts, nuts, etc. These are put together into subassemblies such as generators, dashboard instruments, etc., which are then used to build more complicated subassemblies such as engines, bodies, etc. This process of subassembly has three significant advantages [7, 6]: *reliability*, *efficiency* and *variety*. Subassembly processes are reliable, since elimination of defective subassemblies can be done at each assembly stage. A defective subassembly produced at one stage will not be built into an assembly at the next stage. Also, with regard to efficiency, subassemblies at a stage can be carried out simultaneously. This greatly speeds up an entire assembly process, compared to sequential step-by-step assembly processes where parts are put together only one at a time. Finally, a common subassembly can be used in a variety of different assemblies, or can be used repeatedly in many places in an assembly.

An example subassembly process in biological systems is found in the self-assembly of bacteriophages. Figure 1 shows the simplified pathway for assembly of a type of bacteriophage, T4, which infects the bacterium *E.coli* strain B [2, 5]. The morphogenetic pathway clearly shows formation of a subassembly; a head and a tail form a head-tail subassembly, and this subassembly is then put together with tail fibers. For simplicity, let us write the assembly tree in Figure 1 in a list representation $(F(TH))$, where F, T and H represent tail fiber, tail and head, respectively. A question arises immediately: how did nature find the fixed subassembly sequence $(F(TH))$ out of all possible subassembly sequences? In particular, why not the other possible sequence, $((FT)H)$?

As illustrated in the above example, our interest in this paper is to find out how a particular subassembly sequence is generated in randomized assembly. We will approach the problem by extending

our work on evolutionary design of mechanical conformational switches [16]. An extended design of mechanical conformational switches will be described which can encode several subassembly sequences. The concept of defect will be introduced to randomized assembly of mechanical parts. A genetic algorithm searches the space of parameterized switch designs in order to find the conformational switches that maximize the assembly rate of a set of parts held together with the switches. Several examples are presented and discussed where a particular subassembly sequence emerges via genetic search by simply maximizing the yield of a desired assembly.

1.3 Previous work

1.3.1 Computational models of bacteriophage assembly

For decades, biologists have studied the mechanism of bacteriophage assembly [2, 5], and developed several computational models. Thompson and Goel [17, 18, 7] developed a computer model that simulates the assembly and operation of bacteriophage T4. A simplified model of a virus is constructed from building blocks which are abstractions of the protein molecules. These building blocks are augmented finite automata that can move randomly in their environment and bond to the other blocks. State transition rules of a block specify how bonds can form and how conformational changes propagate within the block. The same approach has been used to model protein biosynthesis [8, 7].

Berger *et al.* [1] developed a local rule theory for self-assembly of icosahedral virus shells. They assume that identical protein subunits take on a small number of distinct conformations. The local rules then specify, for each conformation, which other conformations it can bind to and the approximate interaction angles, interaction length, and torsional angles that can occur between them. By following this local information, the subunits form a closed icosahedral shell with the desired symmetry.

1.3.2 Mechanical model of conformational switches

In the above work, conformational switches are simply predetermined numerical relationships. In contrast, Penrose [14], suggested several designs of mechanical conformational switches that are used in devices that "self-reproduce". These conformational switches cause a bond at one location to break a bond existing at another location or prevent a bond from occurring at another location. When the correct number and arrangement of sub-devices are linked, the conformational switches cause the entire chain to cleave into two copies of the original self-reproducing device in a process akin to cell division.

Another example is found in Hosokawa *et al.* [10]. They developed triangular parts employing switches realized with movable magnets that allow parts to bond together to form hexagons. The switches allow a part to be either in an active or inactive state. An activated part can bond to an inactivated part, turning the part to an activated state. These parts are assembled in a rotating box randomizer. The amounts of each intermediate subassembly achieved agreed reasonably well with the predicted values obtained by techniques analogous to chemical kinetics.

1.3.3 Randomized assembly of mechanical parts

Moncevicz *et al.* [11, 13, 12] developed a layered palletization technique, where parts are "palletized" by using vibration to convey them over a plastic "pallet" into which are carved an array of relief shapes that trap and orient the flowing parts. The first part is designed such that once a quantity is held in the pallet, it becomes integral with the pallet for the purpose of palletizing a quantity of the second part. Therefore, the second part palletization actually assembles the second part to the first part. Since many part insertions occur simultaneously, a very high assembly rate can be achieved.

Yeh and Smith [20] used a (non-layered) palletization technique similar to [13, 12], to assemble microstructures. They fabricated trapezoidal galium arsenite (GaAs) blocks and a Si wafer with trapezoidal holes. Assembly is then done by releasing the GaAs blocks in a carrier fluid (ethanol) and dispensing the fluid over the Si wafer. Cohn *et al.* [3] experimented with the self-assembly of small hexagonal parts (1 *mm* in diameter) by placing a quantity of them on a slightly concave diaphragm that was agitated with a loudspeaker.

1.3.4 Models of subassembly processes in biological assembly

Here, we note two important investigations on subassembly processes in biological assembly. Due to the relevance to our work, these are described in some detail.

Crane [4] has provided a lucid discussion of the advantages of subassembly processes in the construction of complicated structures from elementary subunits. One aspect of his discussion has to do with scheduling the assembly process into a series of stages, or subassembly processes. In his presentation, we start with 1,000,000 identical elementary units. Our objective is to build structures of 1,000 unit length (deca-deca-decamers) using two different subassembly processes. The first subassembly process consists of three stages:

1. Ten elementary units are put together to form 100,000 subassemblies of 10 unit length, or 100,000 decamers.

2. Ten of the decamers are put together to form 10,000 subassemblies of 100 unit length, or 10,000 deca-decamers.

3. Ten of the deca-decamers are put together to form 1,000 final assemblies of 1,000 units long, or 1,000 deca-deca-decamers.

The second subassembly process is to join one thousand elementary units together in a stage. He assumed there was a defect probability that a unit was added wrongly causing the subassembly to become defective. He further assumed the defective subassemblies could not be incorporated into a subassembly at the next stage, and the elementary units in the defective subassemblies were completely wasted. Assuming the same defect probability of 1% at each subassembly stage, the first three-stage subassembly process gives 739 deca-deca-decamers[1] out of a possible 1,000. On the other hand, the second subassembly process with the same defect probability produces only 0.0432 deca-deca-decamer[2]. It is clear that the first three-stage subassembly process is far more efficient than the second one-stage subassembly process.

Following Crane's work, Rosen [15] posed the following question: how can we choose the number of subassembly stages, and the number of elementary units to be put together at each stage, in such a way as to maximize the yield of the desired assembly at the last stage? By extending Crane's subassembly model, he showed the above problem could be formulated as an integer programming problem[3]:

$$\text{maximize} \quad (1-q)^{r_1+r_2+\ldots+r_N}\left(\frac{M}{L}\right) \tag{1}$$

$$\begin{aligned} \text{subject to} \quad & L = r_1 \cdot r_2 \cdot \ldots \cdot r_N \\ & N \geq 0; \quad N \in \mathbf{Z} \\ & r_i \geq 0; \quad r_i \in \mathbf{Z}; \quad i \in \{1, 2, \ldots, N\} \end{aligned} \tag{2}$$

where q, M and L are defect probability, the number of elementary units in the initial pool, and the number of elementary units in a desired assembly, respectively. N is the number of subassembly stages and r_i is the number of subassemblies produced at the $(i-1)$-th stage, which are incorporated into a subassembly at the i-th stage[4]. Since q, M and L are positive constants, and $(1-q) \leq 1$, The above problem is equivalent to minimizing the exponent $r_1 + r_2 + \ldots + r_N$ under the same constraints. The Unique Factorization Theorem[5] in elementary number theory shows that the factorization of L into prime numbers has the property that sum of the factors is minimal. In the case of $L = 1000$, the prime factor decomposition is:

$$L = 1000 = 2 \cdot 2 \cdot 2 \cdot 5 \cdot 5 \cdot 5 \tag{3}$$

[1] Obtained by $(1-0.01)^{10+10+10}(1,000,000/1,000) = 739$; see Appendix A for more detail.

[2] Similarly, by $(1-0.01)^{1,000}(1,000,000/1,000) = 0.0432$; see Appendix A for more detail.

[3] The derivation is found in Appendix A

[4] In the Crane's first subassembly process, for example, $q = 0.01$, $M = 1,000,000$, $L = 1,000$, $N = 3$, and $r_1 = r_2 = r_3 = 10$.

[5] The proof is found in Appendix B

which gives $N = 6$. The corresponding yield is:

$$(1 - 0.01)^{2+2+2+5+5+5} \left(\frac{1,000,000}{1,000} \right) = 810 \tag{4}$$

which is larger than 739, the yield by Crane's first subassembly process. Note that the theorem says nothing about the order of factorization; it makes no difference how we distribute the r_i among the subassembly stages. In this case, for example, $(r_1, r_2, r_3, r_4, r_5, r_6) = (2, 2, 2, 5, 5, 5)$ and $(r_1, r_2, r_3, r_4, r_5, r_6) = (2, 5, 2, 5, 2, 5)$ are equivalent in terms of the yield of the final assemblies.

The importance of Rosen's work lies in the fact that he formulated the problem of finding the best subassembly schedule as an optimization problem, and provided a solution in closed form. There are, however, several points to be generalized to make his model more realistic. First, the defect probability q should not be the same for each subassembly stage. This generalization, however, yields another integer programming problem, which in general cannot be solved in a closed form:

$$\text{maximize} \quad (1 - q_1)^{r_1}(1 - q_2)^{r_2} \ldots (1 - q_N)^{r_N} \left(\frac{M}{L} \right) \tag{5}$$

$$\begin{aligned} \text{subject to} \quad & L = r_1 \cdot r_2 \cdot \ldots \cdot r_N \\ & N \geq 0; \quad N \in \mathbf{Z} \\ & r_i \geq 0; \quad r_i \in \mathbf{Z}; \quad i \in \{1, 2, \ldots, N\} \end{aligned} \tag{6}$$

The second point for generalization is that the model should allow subassembly processes which involve subassemblies with non-equal numbers of subunits. For instance, it should be possible to consider the subassembly process which produces a 1,000-mer out of five 100-mers and four 125-mers at the final stage. Finally, the model should be extended such that it can express the simultaneous construction of subassemblies. It should, for example, be possible to form a 100-mer out of ten 10-mers *as soon as* the first ten 10-mers are produced, *not after* all the elementary units are assembled into 10-mers.

In the following sections, we will discuss our approach, genetic optimization of parameterized conformational switch designs, which incorporates the above generalizations and finds the conformational switch design which encodes the best subassembly sequence in terms of the yield of the final assembly.

2 Conformational switch model for subassembly generation

2.1 One-dimensional conformational switch

We extend our previous design of a one-dimensional mechanical conformational switch (a simple sliding bar mechanism) [16]. The extended design allows a part which cannot bind to another part *before* the conformational change, to form a bond *after* the conformational change. In order to realize this function in a mechanically realistic way, we introduce a new gadget called the *minus device*. Figure 2(a) shows the operation of a minus device. It consists of two short sliding bars that constitute the shapes of two bonding sites, and one inner sliding block that connects them. The left and right pictures in Figure 2(a) show the minus device *before* and *after* conformational change, respectively. Before conformational change, the right sliding bar *cannot* be pushed in due to the position of the inner sliding block. After conformational change induced by pushing in the left sliding bar, the right sliding bar *can* be pushed in since pushing in the left sliding bar causes the inner block to slide down, leaving space for the right sliding bar to slide left. For simplicity, we will often use the abstracted representation of the device, a minus sign with an arrow, shown in Figure 2(b). The direction of the arrow indicates that causality of conformational change. A right-pointing arrow indicates the conformational change is induced by pushing the *left* sliding bar and *vice versa*. Note that in the abstract representation, the right sliding bar after conformational change is drawn as the "pushed-in" state, representing that the bar is "free" to be pushed in.

Figure 2(c) shows how interaction with another part induces conformational change in a minus device. Note that this is a conformational switch design for one-dimensional randomized assembly, where parts can be assembled in only one direction, say horizontally. In Figure 2(c), therefore, one can place a part

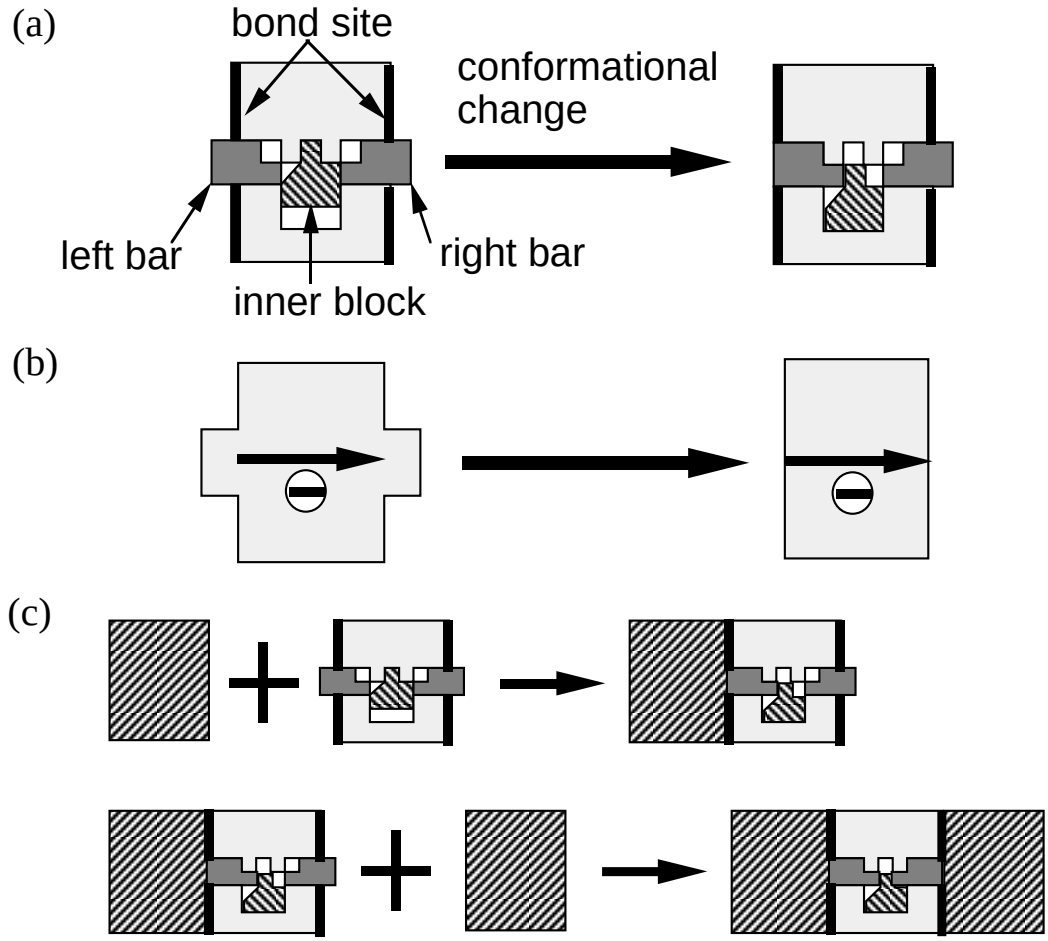


Figure 2: one dimensional conformational switch with a minus device

on the left or the right of another part, but *not* on the top or bottom. However, the model could easily be extended to the two-dimensional case. Note also that conformational change realized by a minus device is *unidirectional* and *irreversible*. In the top picture of Figure 2(c), for instance, no conformational change is induced if the hatched rectangular part is placed on the *right* of the part with the minus device (unidirectionality). Also, since the minus device is not "spring-loaded", it is impossible to reverse the conformational change (irreversibility). This implies that once a part changes its conformation and a bond is formed as a result, it will *never* be destroyed, *i.e.* no detaching is possible.

The actual conformational switch design we use for subassembly generation consists of a "two-digit" bonding site as shown in Figure 3. The two-digit bonding site increases the number of possible shapes it can take, which in turn increases the number of subassembly sequences it can encode[6].

A *bond configuration* is a pair of integers (a_1, a_2), which describes the shape of a bonding site. Each component of a bond configuration takes a positive value if the corresponding "digit" of the bonding site has convex shape, a negative value if the the digit is concave, and zero if it is flat. Examples of bond configurations and the corresponding shapes of bonding sites are shown in Figure 4.

When bonding sites of two parts meet, they form either 1) a stable bond, 2) an unstable bond, or 3) no bond. The occurrences of these cases depend on the shape of the two bonding sites, or equivalently,

[6]To determine the number of "digits" necessary to encode a given assembly tree is an interesting coding problem and will be discussed in Section 5.1.

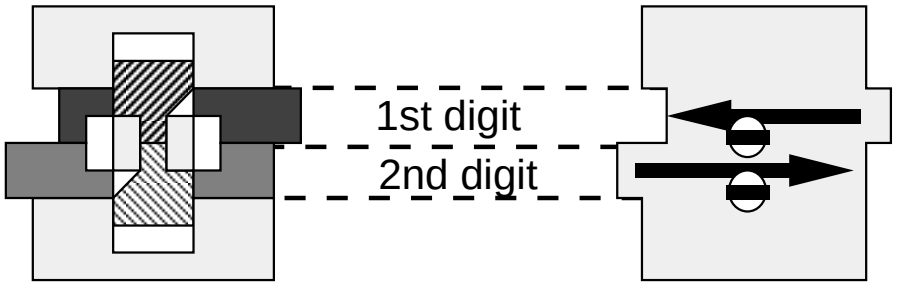


Figure 3: one dimensional conformational switch with "two-digit" bonding site

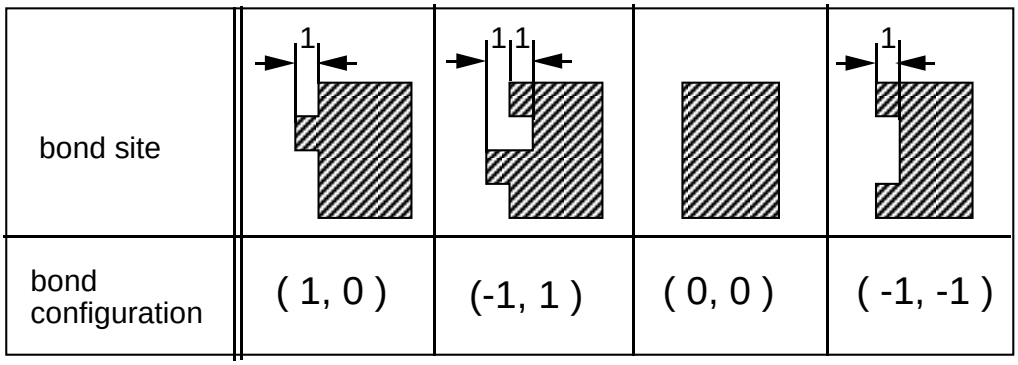


Figure 4: examples of bond configurations and the corresponding bond shapes (of the left bond site)

the bond configurations of the sites. Let (a_1, a_2) and (b_1, b_2) be bond configurations of two bonding sites contacting each other. These sites form a stable bond if they are complementary to each other:

$$a_1 + b_1 \leq 0 \quad \text{and} \quad a_2 + b_2 \leq 0 \tag{7}$$

and form an unstable bond if they are fairly complementary:

$$(a_1 + b_1 = 1 \quad \text{and} \quad a_2 + b_2 \leq 0) \quad \text{or} \quad (a_1 + b_1 \leq 0 \quad \text{and} \quad a_2 + b_2 = 1) \tag{8}$$

If none of the above apply, the two bonding sites are not complementary and therefore no bond is formed. Figure 5 shows some examples of each of the three cases. An unstable bond induces conformational change of the involved bonding sites. After the conformational changes, a stable bond is formed if the condition (7) is satisfied. Otherwise, no bond is formed.

Ambiguous situations may arise when conformational change can propagate in both directions such as the case shown in Figure 6(a). In such cases, we assume propagation goes upstream as defined in Figure 6(b) (priority to *upstream propagation*).

2.2 One-dimensional randomized assembly with defect

As in our previous work [16] we will simulate the following *robot bin-picking metaphor* in our simulation of one-dimensional randomized assembly:
Assume a random assortment of parts in a (one-dimensional) bin (Figure 7(a)).

Step 1: The robot arm #1 randomly picks up a part from the bin. Then, robot arm #2 randomly picks up another part from the bin (Figure 7(b)).

Step 2: The two parts are pushed against each other, possibly causing formation of a bond (Figure 7(c)).

Step 3: The parts are randomly returned to the bin (Figure 7(d)), possibly as an assembly.

The steps 1-3 are repeated until pre-specified conditions are satisfied (*e.g.* repeat for a specified number of iterations, repeat until the number of parts decreases below a limit, *etc.*). It is assumed that the parts do not change their orientations, so in general, AB and BA are two distinct assemblies.

We follow Crane's assumption of defects [4] during the above randomized assembly process. Namely, we assume 1) with a certain probability two subassemblies (or two parts) are put together incorrectly, causing the resulting subassembly to be defective, and 2) the defective subassemblies cannot be incorporated into the subsequent subassemblies, so the parts in the defective subassemblies are completely wasted. In the scenario above, a defect could occur at step 2 with a certain probability if two parts picked can form a bond[7], with the defective subassembly being returned to the bin. If the defective subassembly is chosen at step 1 in subsequent iterations, no bond is formed at the step 2, *regardless of* the bond configurations of the two mating sites. In this manner defective subassemblies "waste" an iteration of the robot bin-picking.

Note that the above *robot bin-picking metaphor* realizes the three generalizations to Rosen's subassembly model described in Section 1.3.4. It is possible to specify different defect probabilities for different combinations of parts or bond configurations. Also, since the scenario does not formulate subassembly sequences explicitly, rather they are assumed to emerge by searching the space of possible conformational switch designs, *any* subassembly sequence can be realized if it can be encoded by the conformational switch model[8], including the ones which involve subassemblies with non-identical numbers of parts. Finally, even though there are only two robot arms (as opposed to many robot arms for parallel assembly), the scenario can simulate the simultaneous construction of several subassemblies since it is globally synchronized based on mating of two subassemblies, rather than completion of a subassembly stage.

[7] If they cannot form a bond, no defect occurs and they are simply returned to the bin.

[8] Again, this raises the question of which subassembly sequences can be encoded by a particular conformational switch model. Some discussion on this issue is found in Sections 4.4 and 5.1.

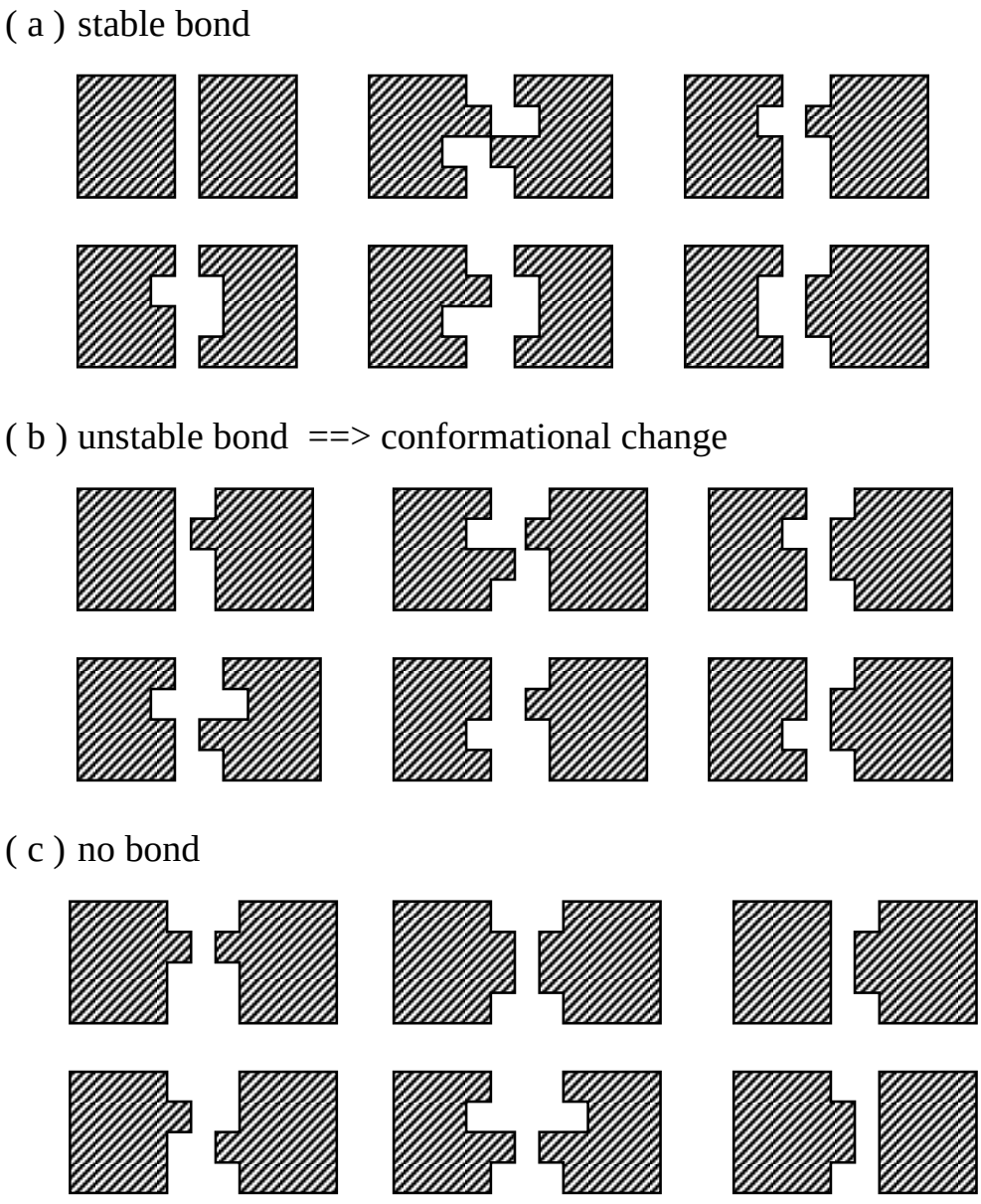


Figure 5: examples of three types of bonding

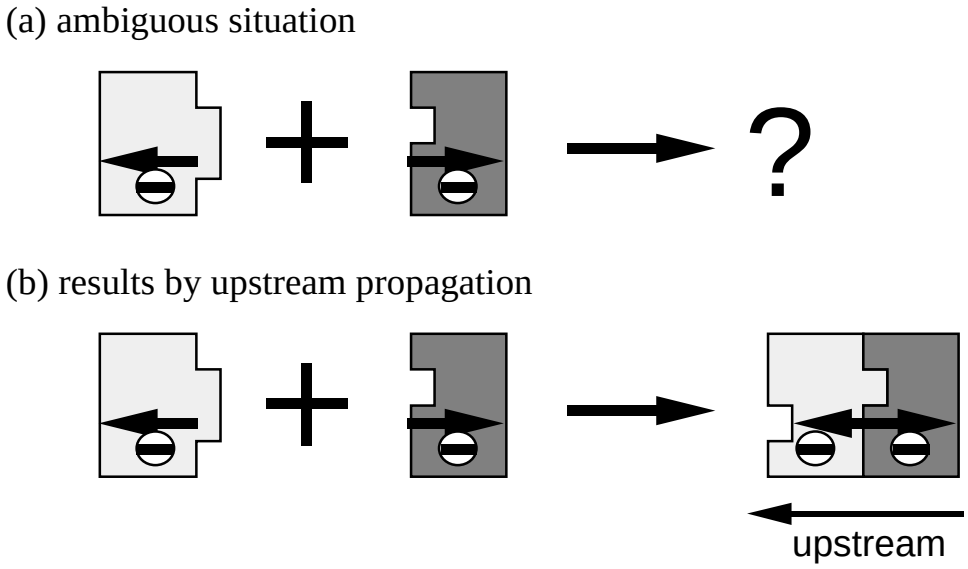


Figure 6: priority to upstream propagation

3 Genetic algorithm optimization

3.1 Genetic algorithms

Genetic algorithms (GAs) are a search technique in which points in the design space are analogous to organisms subject to a process of natural selection, or "survival of the fittest" [9]. GAs model reproduction in populations of an encoded representation of points in design space – genetic "chromosomes" – over generations. In a given generation, the quality of a chromosome (or a point in design space) is measured based on a fitness function, and highly-fit chromosomes have higher chances to be selected for reproduction. Two "parent" chromosomes selected for reproduction are mated through genetic crossover, resulting in two offspring chromosomes which are likely to inherit good "genes" from their parents. Many generations of such selection and mating will produce a highly-fit population of chromosomes, *i.e.* better designs.

3.2 Formulation of GA search

We are interested in designing conformational switches that, when randomly assembled, maximize the yield of a desired assembly. More precisely, the problem is stated as follows:

Given: the total number of parts, and the number of each kind of part in the bin (in other words, the initial state of the bin), and defect probabilities.

Find: the optimal design of conformational switches that maximizes the yield of a desired set of parts in the process of randomized assembly.

We are designing not only the bar mechanisms, but also the initial bond configurations. The above problem is formulated as search by parameterizing the design of the conformational switches. A genetic algorithm is used to search the parameter space of possible switch designs. Note that we are searching the space of possible conformational switch designs, *not* the space of subassembly sequences. There is no explicit formulation of subassembly sequences in the above problem. Rather, an optimal subassembly

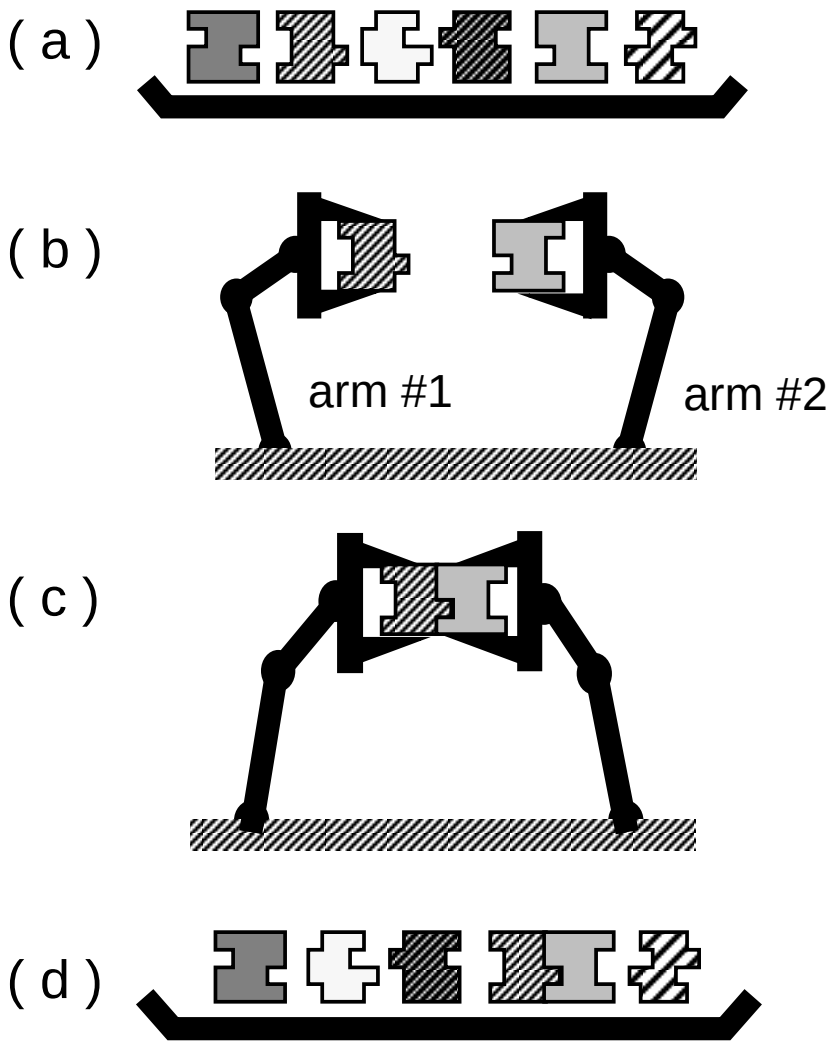


Figure 7: robot bin-picking metaphor

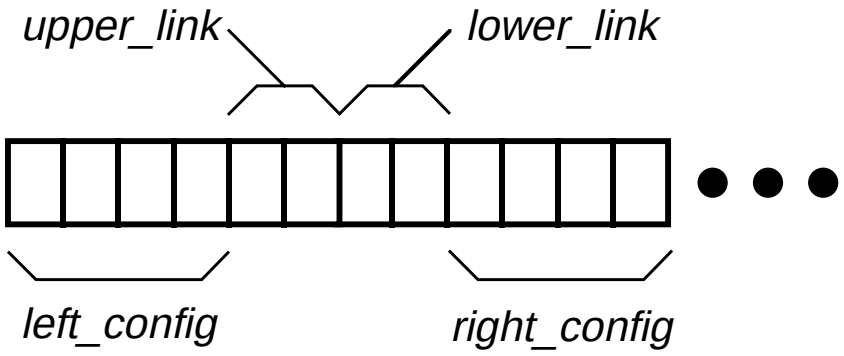


Figure 8: bit assignment of a chromosome

sequence *emerges* due to a particular design of conformational switches that maximizes the yield of the desired assembly.

The one-dimensional conformational switch of a part is uniquely specified in terms of four parameters: *left_config*, *right_config*, *upper_link* and *lower_link*. *Left_config* and *right_config* are the initial bond configurations of the left bond site and the right bond site, respectively. *Upper_link* and *lower_link* are variables that specify the existence of conformational links (minus devices) in a part, each of which takes one of the values *LEFT*, *RIGHT* or *NONE*. If *upper_link* is *LEFT*, there is a conformational link between the "first digits" of the bond sites, with the direction pointing to the left bond site. If *upper_link* is *RIGHT*, there is a conformational link between the first digits of the bond sites pointing to the right bond site. If *upper_link* is *NONE*, there is no conformational link that connects the first digits of the bond sites, so the first digits cannot undergo any conformational change. Similarly, *lower_link* specifies the existence of the conformational link between the second digits of the bond sites. Note that the conformational links are independent of each other, so the second digits can change their conformation, for instance, even though the first digits cannot.

A chromosome used in genetic search is a binary string that encodes the above design parameters for all kinds of parts in the bin. For the examples in the next section, two bits are assigned to each component of bond configuration[9], therefore each of *left_config* and *right_config* occupies four bits. In addition, two bits are used for each of *upper_link* and *lower_link*. The location of these bits on a chromosome is shown in Figure 8.

For each component of *left_config* and *right_config*, the first bit corresponds to sign, with plus being 0 and minus being 1, and the second bit corresponds to the absolute value[10]. If $left\_config = (1, 0)$, for example, the corresponding four bits are 0100 or 0110. The first bit for *upper_link* and *lower_link* represents the direction of the conformational link, with right ($\rightarrow$) being 0 and left ($\leftarrow$) being 1. The second bit represents the existence of the link. The bit is 1 if the link exists, and 0 if there is no link. The first bit is ignored if the second bit is 0. For instance, the corresponding two bits for $lower\_link = NONE$ is either 10 or 00. Figure 9 shows an example of a parameter encoding of a part. Since twelve bits are necessary for one part, a chromosome that encodes n kinds of parts has length $12n$.

[9]Recall a bond configuration is a pair of integer (a_1, a_2).

[10]This implies that each component of *left_config* and *right_config* can only take values $\{-1, 0, 1\}$.

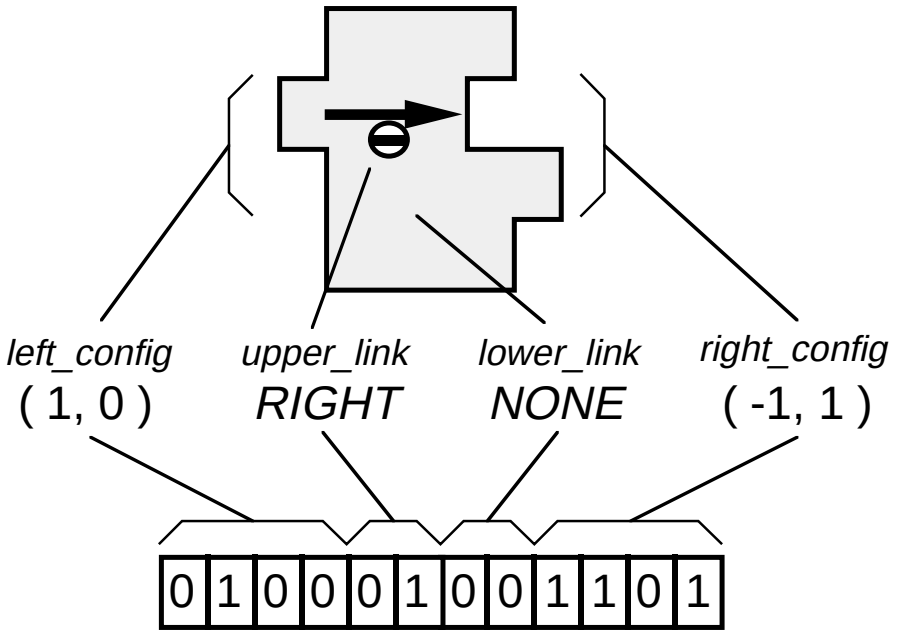


Figure 9: an example of parameter encoding of a part

3.3 Reinforcement evaluation

A chromosome is evaluated by decoding the bit string representation (genotype) to a part design (phenotyope), and by running the robot bin-picking simulation with the decoded part design. The fitness of a chromosome (*i.e.* the suitability of conformational switch designs and their initial configurations) is simply the number of desired assemblies produced after some number of iterations of the robot bin-picking simulation. In order to reduce stochastic error due to random picking during the simulation, the actual fitness is measured as an average of a pre-specified number of such bin-picking runs.

For further reduction of stochastic error involved in the fitness evaluation, we introduce a new scheme called *reinforcement evaluation*. In this scheme, the best chromosome in the population is evaluated again at the end of each generation. The fitness of the best chromosome is then re-calculated to "reinforce" the evaluation:

$$\bar{f}_{new} = \frac{n \cdot \bar{f}_{old} + f}{n+1} \tag{9}$$

where $\bar{f}_{old}$ and $\bar{f}_{new}$ are the fitness values before and after reinforcement, f is the fitness value obtained by the additional evaluation, and n is the number of times the chromosome has been evaluated. This ensures that only good chromosomes that are carried over generations have more chances to be evaluated many times, reducing stochastic error of their fitness values.

4 Examples

This section describes two examples of genetic optimization of one-dimensional conformational switches. The GA in the following examples uses a crowding population [9] with 10% replacement per generation, based on hamming distance between chromosomes, fitness proportionate (roulette wheel) selection [9], linear fitness scaling [9] with scaling coefficient = 2.0, and unless otherwise specified, crossover probability = 0.9 and mutation probability = 0.03. Also, reinforcement evaluation is performed as described in the previous section.

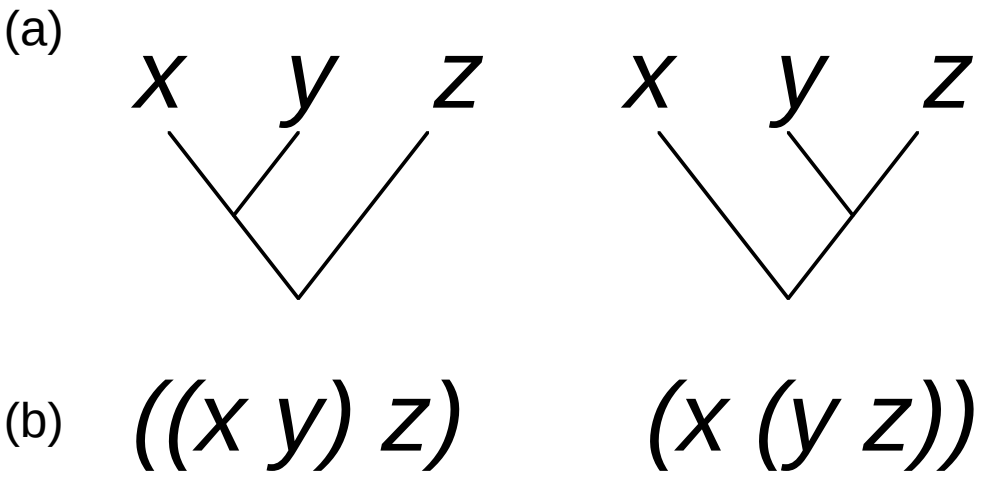


Figure 10: representations of subassembly sequences: (a) binary tree representation, and (b) list representation. x, y and z are subassemblies.

Before proceeding to the results, let us define our notion of subassembly sequences in one-dimensional assembly. We define a *subassembly* to be a set of one or more parts connected together. In particular, a part is a subassembly. A *subassembly sequence* is a sequence in which *two* subassemblies are put together to produce a final assembly. According to the above definition, *any* fixed (non-ambiguous) subassembly sequence of a one-dimensional assembly can be represented uniquely by a binary tree (Figure 10(a)) or by a list representation (Figure 10(b))[11] It is, however, often the case that the assembly sequence encoded by a conformational switch design is ambiguous, or under-specified. We use the notation $\{u\}$ if (and only if) the subassembly sequence to build a subassembly u is not specified. In the case where $u = xyz$, for instance, $\{xyz\}$ indicates that the three subassemblies x, y and z can be put together in any order, i.e. either $((xy)z)$ or $(x(yz))$.

4.1 Three part randomized assembly

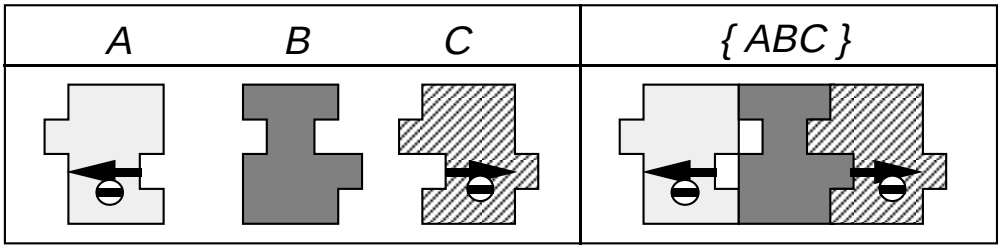
The first example is a three part randomized assembly as described in Section 2.2. The initial bin contains a random mixture of three types of parts, part A, part B and part C, and the design objective is to maximize the yield of the assembly ABC. The number of ABC's in the bin is counted after 700 iterations of Steps 1-3 in Section 2.2. At each evaluation of a chromosome, an average is taken for the count of ABC's over 50 such runs. The GA runs described in this section have population size of 300 and the number of generations is 200. In the following results, $\mathbf{n}_0 = (n_A(0), n_B(0), n_C(0))$ is the vector of the initial numbers of parts A, B and C in the bin, and $\mathbf{q} = (q_{AB}, q_{BC})$ is the vector of defect probabilities of the bonds between AB and BC, respectively[12]. Note that there are only two possible subassembly sequences in this example: $((AB)C)$ and $(A(BC))$.

Figure 11 shows the best designs of conformational switches obtained from GA runs with $\mathbf{n}_0 = (10, 10, 10)$, and with (a) $\mathbf{q} = (0.0, 0.0)$, (b) $\mathbf{q} = (0.2, 0.0)$ and (c) $\mathbf{q} = (0.0, 0.2)$. In all three cases, the parts are designed such that *only* ABC can form through random mating (*e.g.* CAB is not possible). During assembly of ABC, however, no conformational links are actually used, *i.e.* no parts undergo conformational changes. This implies that none of the three designs specifies a fixed subassembly sequence: an ABC can assemble in any of the two possible sequences, $((AB)C)$ or $(A(BC))$. In other words, these

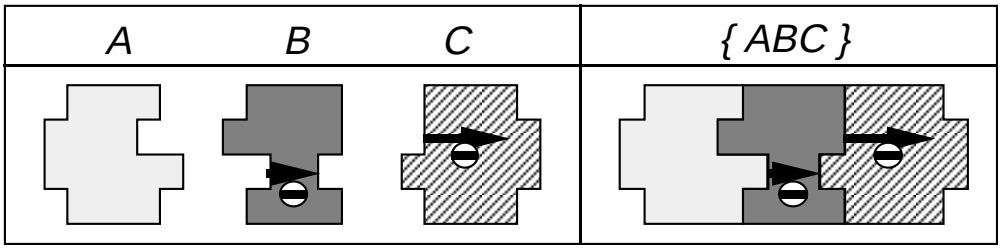
[11] Since we are dealing with one-dimensional assembly, the above binary tree representation can specify *both* a final assembly *and* the order of assembly, whereas in mechanical assembly an *assembly tree* usually specifies only the order of assembly.

[12] We assume defect probability of a bond depends only on the parts associated to the bond. In particular, we assume $q_{AB} = q_{A(BC)}$ and $q_{BC} = q_{(AB)C}$.

(a) ***q*** = (0.0, 0.0); fitness = 9.99; n = 1



(b) ***q*** = (0.2, 0.0); fitness = 8.09; n = 1



(c) ***q*** = (0.0, 0.2); fitness = 8.01; n = 1

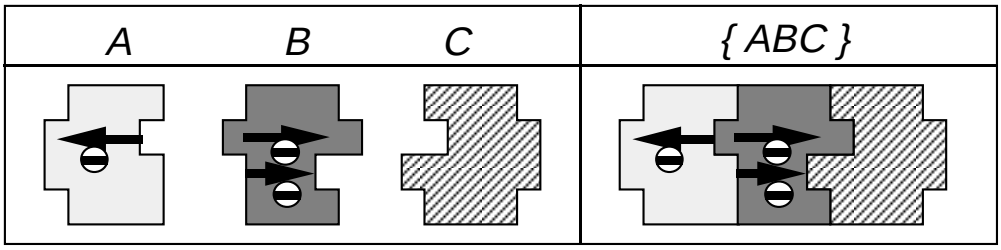


Figure 11: best designs with $\mathbf{n}_0 = (10, 10, 10)$

conformational switch designs encode $\{ABC\}$.

On the other hand, A non-ambiguous subassembly sequence emerges in the case where there are more part B's than part A's and part C's. Figure 12 shows the resulting switch designs in the case $\mathbf{n}_0 = (10, 20, 10)$. For $\mathbf{q} = (0.0, 0.0)$ and $\mathbf{q}$=(0.0, 0.2), a part A can bind to a part B *only after* part B changes its conformation, which is triggered by the binding of part C (see Figures 12(a) and 12(c)). The formation of an assembly ABC, therefore, takes place through the following two-step "reactions":

$$B + C \rightarrow B'C \tag{10}$$

$$A + B'C \rightarrow AB'C \tag{11}$$

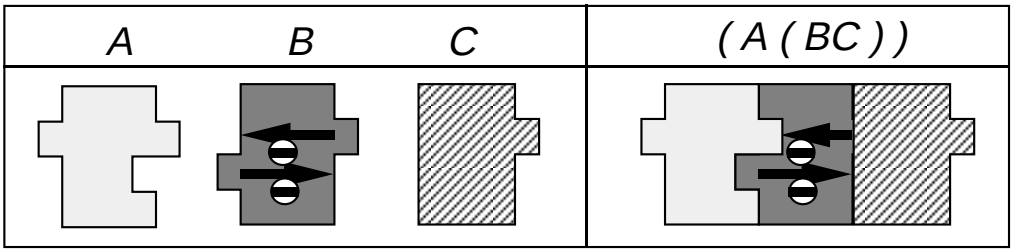
where B' represents a part B after conformational change. Since no other reactions are possible, an ABC assembles *only* in the fixed subassembly sequence $(A(BC))$. On the other hand, $((AB)C)$ is encoded in the best design with $\mathbf{q} = (0.2, 0.0)$ as shown in Figure 12(b). In this case, a part C can bind to a part B *only after* the part B changes its conformation, which is triggered by binding of a part A:

$$A + B \rightarrow AB' \tag{12}$$

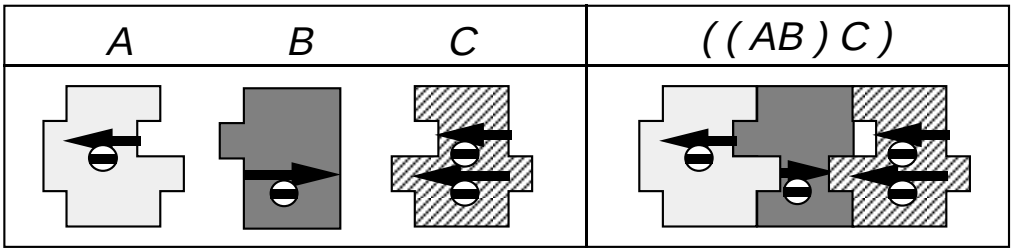
$$AB' + C \rightarrow AB'C \tag{13}$$

Note that only one conformational link is actually used during the assembly of ABC in both cases. The

(a) ***q*** = (0.0, 0.0); fitness = 9.52; n = 1



(b) ***q*** = (0.2, 0.0); fitness = 7.84; n = 9



(c) ***q*** = (0.0, 0.2); fitness = 8.07; n = 2

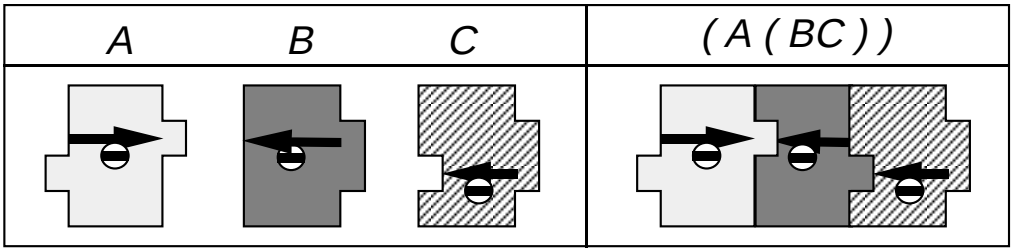
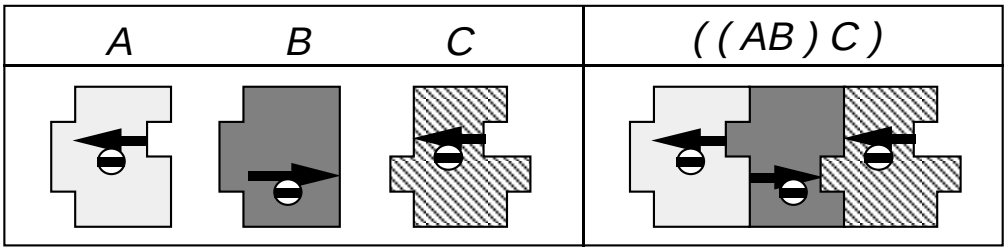
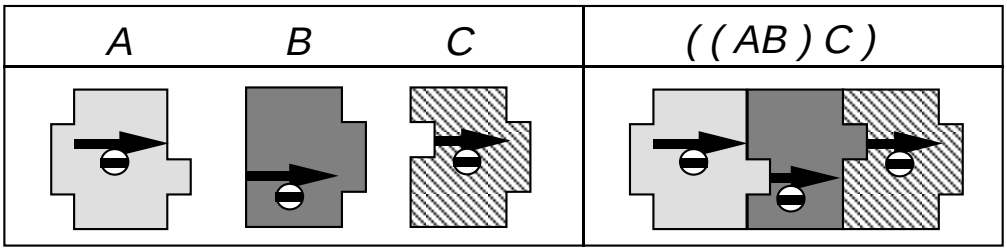


Figure 12: best designs with $\mathbf{n}_0 = (10, 20, 10)$

(a) ***q*** = (0.0, 0.0); fitness = 10.00; n = 11



(b) ***q*** = (0.2, 0.0); fitness = 9.98; n = 2



(c) ***q*** = (0.0, 0.2); fitness = 8.21; n = 1

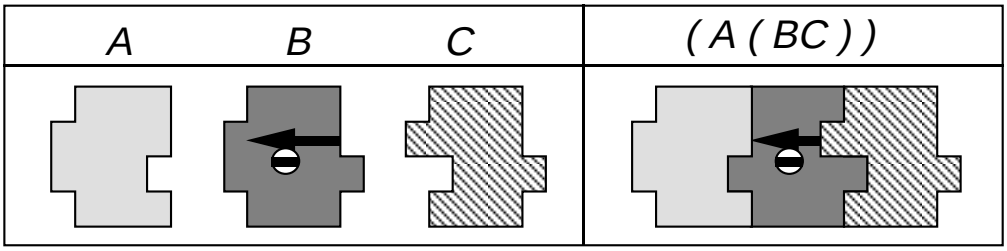


Figure 13: best designs with $\mathbf{n}_0 = (20, 20, 10)$

results of GA runs with $\mathbf{n}_0 = (20, 20, 10)$ are shown in Figure 13. The best designs encode $((AB)C)$ for $\mathbf{q} = (0.0, 0.0)$ and $\mathbf{q} = (0.2, 0.0)$, and $(A(BC))$ for $\mathbf{q} = (0.0, 0.2)$.

4.2 Rate equation analyses of three part randomized assembly

Discrete-time rate equation analyses are done in order to understand the emergence of a particular subassembly sequence in the above example. We generalize the rate equation formulation for defect-free randomized assembly [16] to incorporate the effect of defects in assembly. Given the part designs, their possible "reactions" and defect probabilities, the following recurrence relation describes the randomized assembly process described in Section 2.2:

$$\mathbf{n}(t+1) = \mathbf{n}(t) + \mathrm{A}\mathbf{p}(t) \tag{14}$$

where $\mathbf{n}(t)$ is a vector of the numbers of each possible subassembly (including the defective ones) at iteration t, $\mathbf{p}(t)$ is a vector of probabilities for each possible reaction at iteration t, and A is a matrix of stoichiometric coefficients.

The following two reactions among parts A, B and C are necessary and sufficient to produces an

assembly ABC in the subassembly sequence $((AB)C))$[13]:

$$A + B \rightarrow AB \tag{15}$$
$$AB + C \rightarrow ABC \tag{16}$$

The above reactions can be interpreted as "an A and a B yield an AB with probability 1, and an AB and a C yield an ABC with probability 1." Assuming the defective subassemblies cannot be incorporated into the subsequent subassemblies, the reactions (15), (16) can be generalized to non-zero defect probabilities as follows:

$$A + B \rightarrow (1 - q_{AB})AB + q_{AB}AB_d \tag{17}$$
$$AB + C \rightarrow (1 - q_{BC})ABC + q_{BC}ABC_d \tag{18}$$

where AB_d and ABC_d denote a defective AB and a defective ABC; and q_{AB} and q_{BC} are defect probabilities of bonding between A and B, and between B and C, respectively. The corresponding $\mathbf{n}(t)$, A and $\mathbf{p}(t)$ are, therefore, defined as follows:

$$\mathbf{n}(t) = \begin{pmatrix} n_A(t) \\ n_B(t) \\ n_C(t) \\ n_{AB}(t) \\ n_{AB_d}(t) \\ n_{ABC}(t) \\ n_{ABC_d}(t) \end{pmatrix}, \mathrm{A} = \begin{pmatrix} -1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 1 - q_{AB} & -1 \\ q_{AB} & 0 \\ 0 & 1 - q_{BC} \\ 0 & q_{BC} \end{pmatrix}, \mathbf{p}(t) = \begin{pmatrix} \dfrac{n_A(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \\ \\ \dfrac{n_{AB}(t) \cdot n_C(t)}{s(t) \cdot \{s(t) - 1\}} \end{pmatrix} \tag{19}$$

where $n_A(t)$, $n_B(t)$, $n_C(t)$, $n_{AB}(t)$, $n_{AB_d}(t)$, $n_{ABC}(t)$ and $n_{ABC_d}(t)$ are the numbers of A, B, C, AB, AB_d, ABC and ABC_d at iteration t, respectively, and $s(t)$ is the sum of all the components of $\mathbf{n}(t)$.

Similarly, in the case of $(A(BC))$, the possible generalized reactions are:

$$B + C \rightarrow (1 - q_{CB})BC + q_{BC}BC_d \tag{20}$$
$$A + BC \rightarrow (1 - q_{AB})ABC + q_{AB}ABC_d \tag{21}$$

and the corresponding $\mathbf{n}(t)$, A and $\mathbf{p}(t)$ are:

$$\mathbf{n}(t) = \begin{pmatrix} n_A(t) \\ n_B(t) \\ n_C(t) \\ n_{BC}(t) \\ n_{BC_d}(t) \\ n_{ABC}(t) \\ n_{ABC_d}(t) \end{pmatrix}, \mathrm{A} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \\ -1 & 0 \\ 1 - q_{BC} & -1 \\ q_{BC} & 0 \\ 0 & 1 - q_{AB} \\ 0 & q_{AB} \end{pmatrix}, \mathbf{p}(t) = \begin{pmatrix} \dfrac{n_B(t) \cdot n_C(t)}{s(t) \cdot \{s(t) - 1\}} \\ \\ \dfrac{n_A(t) \cdot n_{BC}(t)}{s(t) \cdot \{s(t) - 1\}} \end{pmatrix} \tag{22}$$

Since both $((AB)C)$ and $(A(BC))$ are possible in the case of $\{ABC\}$, the rate equations for $\{ABC\}$ are constructed by simply merging (19) and (22) together:

$$\mathbf{n}(t) = \begin{pmatrix} n_A(t) \\ n_B(t) \\ n_C(t) \\ n_{AB}(t) \\ n_{AB_d}(t) \\ n_{BC}(t) \\ n_{BC_d}(t) \\ n_{ABC}(t) \\ n_{ABC_d}(t) \end{pmatrix}, \mathrm{A} = \begin{pmatrix} -1 & 0 & -1 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 \\ 1 - q_{AB} & 0 & 0 & -1 \\ q_{AB} & 0 & 0 & 0 \\ 0 & 1 - q_{BC} & -1 & 0 \\ 0 & q_{BC} & 0 & 0 \\ 0 & 0 & 1 - q_{AB} & 1 - q_{BC} \\ 0 & 0 & q_{AB} & q_{BC} \end{pmatrix} \tag{23}$$

[13]Conformational change is ignored in the notation here.

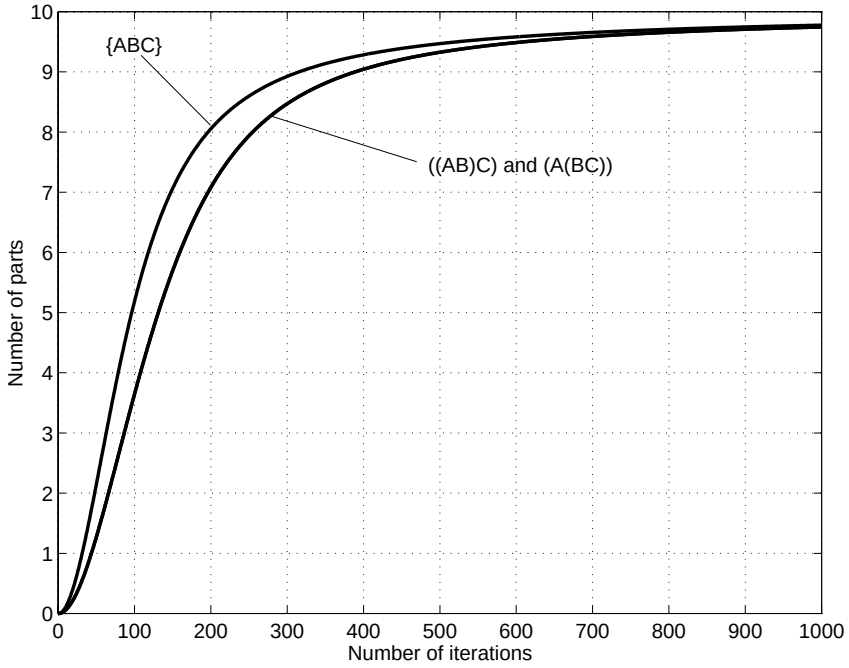


Figure 14: solution with $\mathbf{n}_0 = (10, 10, 10)$ and $\mathbf{q} = (0.0, 0.0)$

$$\mathbf{p}(t) = \begin{pmatrix} \dfrac{n_A(t) \cdot n_B(t)}{s(t) \cdot \{s(t) - 1\}} \\ \dfrac{n_B(t) \cdot n_C(t)}{s(t) \cdot \{s(t) - 1\}} \\ \dfrac{n_A(t) \cdot n_{BC}(t)}{s(t) \cdot \{s(t) - 1\}} \\ \dfrac{n_{AB}(t) \cdot n_C(t)}{s(t) \cdot \{s(t) - 1\}} \end{pmatrix} \tag{24}$$

The equation (14) is numerically solved for different initial conditions and defect probabilities, in order to compare the dynamic behavior of parts (and defective parts) in $((AB)C)$, $(A(BC))$ and $\{ABC\}$. Figures 14, 15 and 16, show the solution with the initial condition $\mathbf{n}_0 = (10, 10, 10)$ and $\mathbf{q} = (0.0, 0.0)$, $\mathbf{q} = (0.2, 0.0)$ and $\mathbf{q} = (0.0, 0.2)$, respectively. The yield of $\{ABC\}$ is slightly better than $((AB)C)$ and $(A(BC))$ in all of the three cases, which matches the results obtained by the GA search[14] shown in Figure 11. For $\mathbf{q} = (0.0, 0.0)$, there is no difference between the solution of $((AB)C)$ and $(A(BC))$. It is observed, however, that the higher defect probability between AB ($\mathbf{q} = (0.2, 0.0)$) slightly favors the subassembly sequence $((AB)C)$ (Figure 15), while $\mathbf{q} = (0.0, 0.2)$ slightly favors $(A(BC))$ (Figure 16).

These trends become more prominent with $\mathbf{n}_0 = (10, 20, 10)$, as shown in Figures 17, 18 and 19. The result of $((AB)C)$ and $(A(BC))$ are identical for $\mathbf{q} = (0.0, 0.0)$, which are better than the yield of $\{ABC\}$. For $\mathbf{q} = (0.2, 0.0)$, the yield of $((AB)C)$ is approximately 5% better than the yield of $(A(BC))$, whereas for $\mathbf{q} = (0.0, 0.2)$, the yield of $(A(BC))$ is approximately 5% better than the yield of $((AB)C)$. The above results support the results by GA runs shown in Figure 12. It should be noted that in the case of $\mathbf{n}_0 = (10, 20, 10)$, the yield of $\{ABC\}$ goes down to about 50% of the maximum possible yield. This rather counter-intuitive drop of the yield is due to the large number of the middle part B, which

[14]Recall the robot bin-picking simulation used in the GA search terminates at 700 iterations.

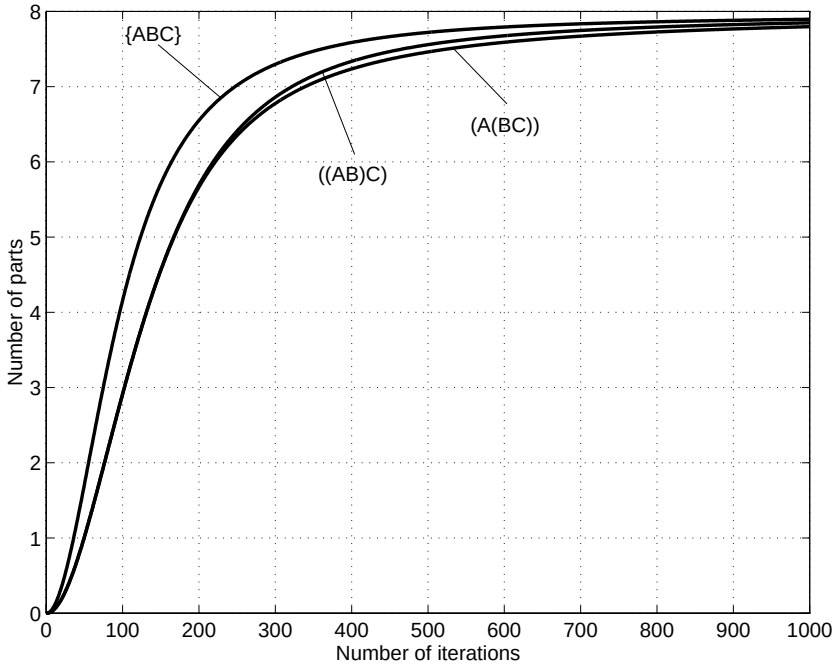


Figure 15: solution with $\mathbf{n}_0 = (10, 10, 10)$ and $\mathbf{q} = (0.2, 0.0)$

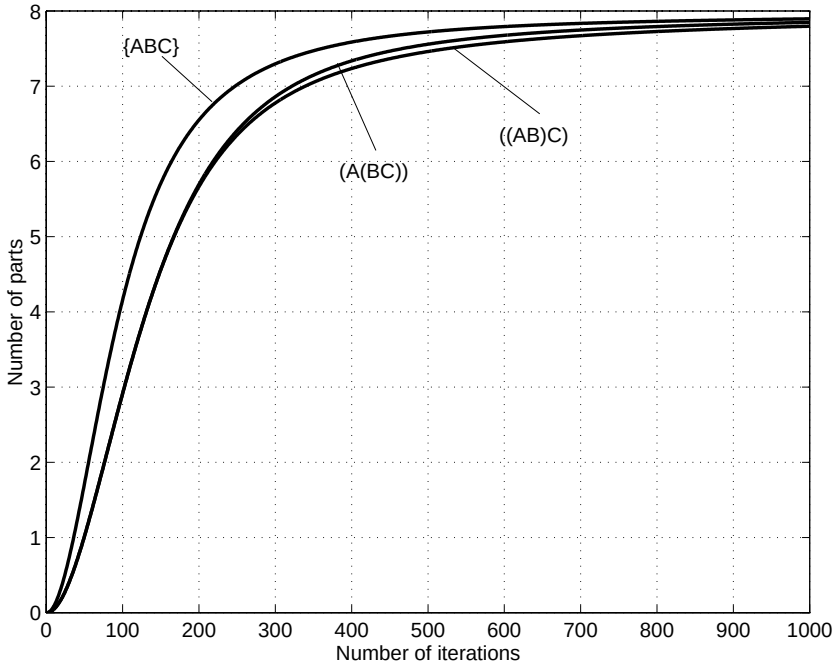


Figure 16: solution with $\mathbf{n}_0 = (10, 10, 10)$ and $\mathbf{q} = (0.0, 0.2)$

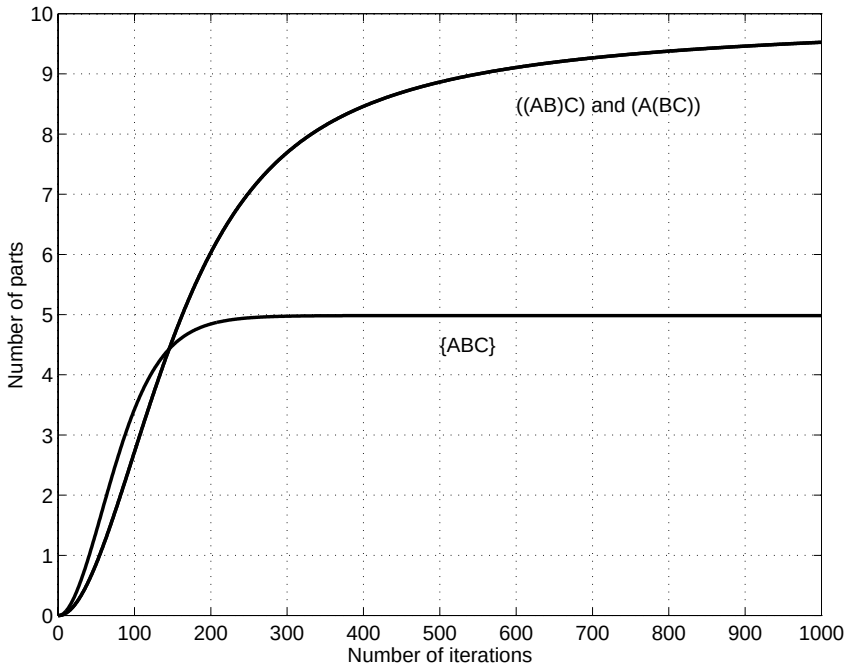


Figure 17: solution with $\mathbf{n}_0 = (10, 20, 10)$ and $\mathbf{q} = (0.0, 0.0)$

produces a large number of AB's and BC's in the early stage of iterations. The excess production of AB's and BC's then causes the shortage of individual C's and A's later on, which are necessary to complete the final assembly ABC from the subassemblies AB and BC. By enforcing the subassembly sequence $((AB)C)$ or $(A(BC))$, this excess production of AB and BC can be avoided. Figures 20, 21 and 22 show the solution of equation (14) with $\mathbf{n}_0 = (20, 20, 10)$ and $\mathbf{q} = (0.0, 0.0)$, $\mathbf{q} = (0.2, 0.0)$ and $\mathbf{q} = (0.0, 0.2)$, respectively. In all cases, $\{ABC\}$ has the highest rate of the desired assembly during the early iterations. However, $((AB)C)$ becomes better approximately after the 250-th iteration and scores the best overall yield. Differences in the final yield between $((AB)C)$ and $\{ABC\}$ are the largest for $\mathbf{q} = (0.2, 0.0)$ and the smallest for $\mathbf{q} = (0.0, 0.2)$. In particular, in the case of $\mathbf{q} = (0.0, 0.2)$, the overall yield is almost identical for $((AB)C)$, $(A(BC))$ and $\{ABC\}$. The GA search in Figure 13 found the optimal solutions (*i.e.* conformational switch designs that encode the optimal subassembly sequence) for $\mathbf{q} = (0.0, 0.0)$ and $\mathbf{q} = (0.2, 0.0)$, and found a suboptimal solution within 1% of the optimal solution for $\mathbf{q} = (0.0, 0.2)$.

4.3 Four part randomized assembly

The second example is a four part randomized assembly. The initial bin contains a random mixture of four types of parts, part A, part B, part C and part D, and the design objective is to maximize the yield of the assembly $ABCD$. The number of $ABCD$'s in the bin is counted after 1400 iterations of Steps 1-3 in Section 2.2. At each evaluation of a chromosome, an average is taken for the count of $ABCD$'s over 50 such runs. The GA runs described in this section have population size of 600 and the number of generations is 900. In the following results, $\mathbf{n}_0 = (n_A(0), n_B(0), n_C(0), n_D(0))$ is the vector of the initial numbers of parts A, B, C and D in the bin, and $\mathbf{q} = (q_{AB}, q_{BC}, q_{CD})$ is the vector of defect probabilities of the bonds between AB, BC and CD, respectively. Figure 23 shows the five non-ambiguous subassembly sequences possible for the four part assembly.

The best designs found by the GA are shown in Figure 24 with $\mathbf{n}_0 = (10, 10, 10, 10)$ and four different defect probabilities: (a) $\mathbf{q} = (0.0, 0.0, 0.0)$, (b) $\mathbf{q} = (0.2, 0.0, 0.0)$, (c) $\mathbf{q} = (0.0, 0.2, 0.0)$ and (d) $\mathbf{q} = (0.2, 0.0, 0.2)$. As in the case of the three part assembly, the parts are evolved such that *only* $ABCD$

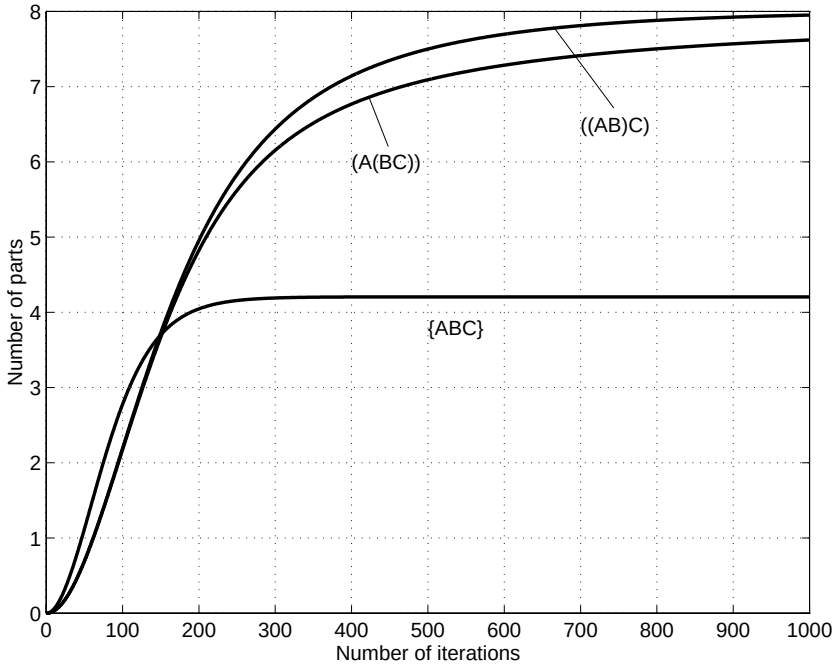


Figure 18: solution with $\mathbf{n}_0 = (10, 20, 10)$ and $\mathbf{q} = (0.2, 0.0)$

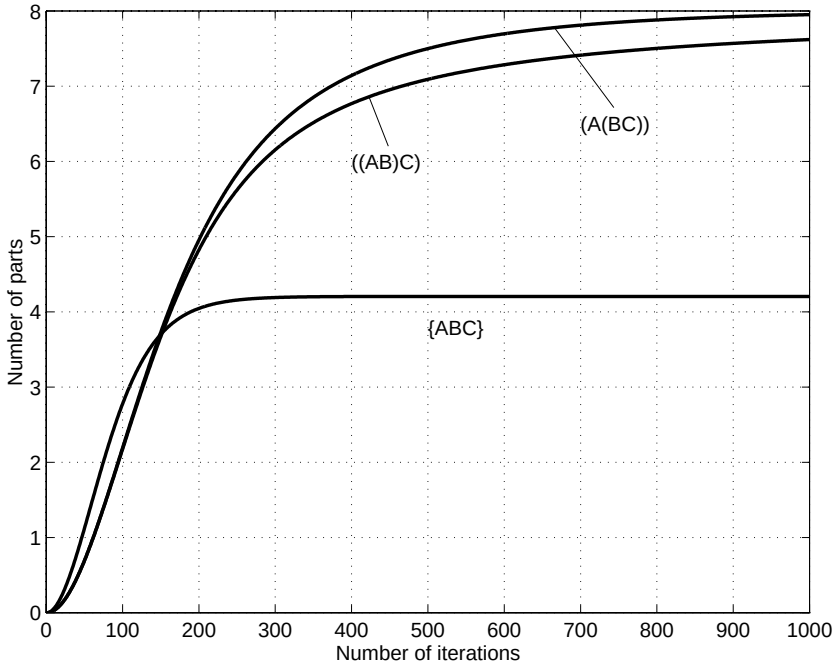


Figure 19: solution with $\mathbf{n}_0 = (10, 20, 10)$ and $\mathbf{q} = (0.0, 0.2)$

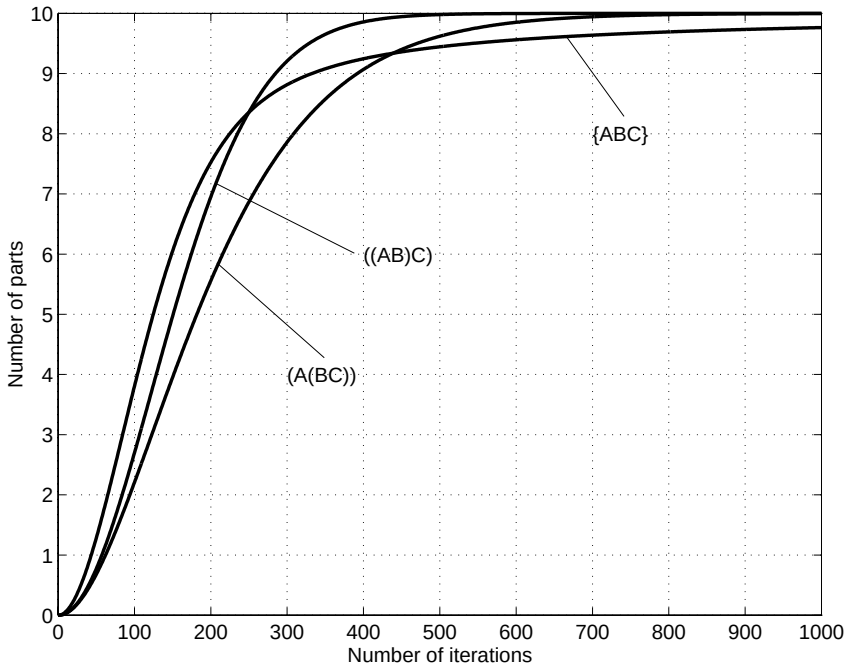


Figure 20: solution with $\mathbf{n}_0 = (20, 20, 10)$ and $\mathbf{q} = (0.0, 0.0)$

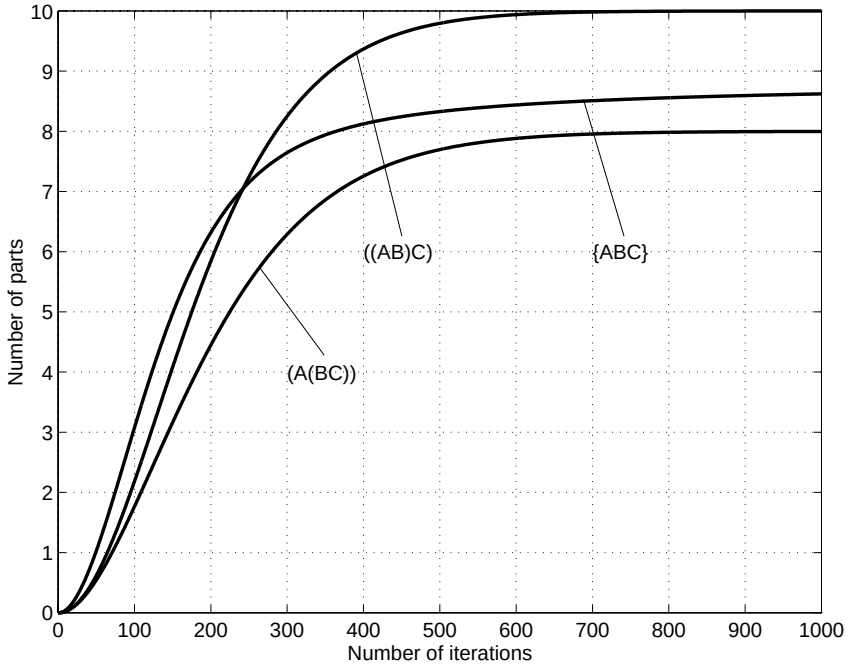


Figure 21: solution with $\mathbf{n}_0 = (20, 20, 10)$ and $\mathbf{q} = (0.2, 0.0)$

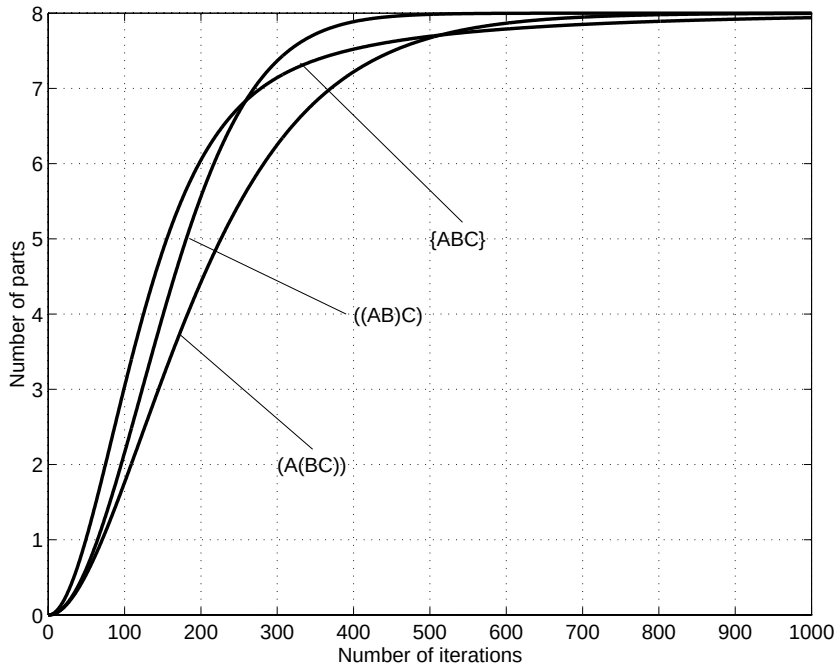


Figure 22: solution with $\mathbf{n}_0 = (20, 20, 10)$ and $\mathbf{q} = (0.0, 0.2)$

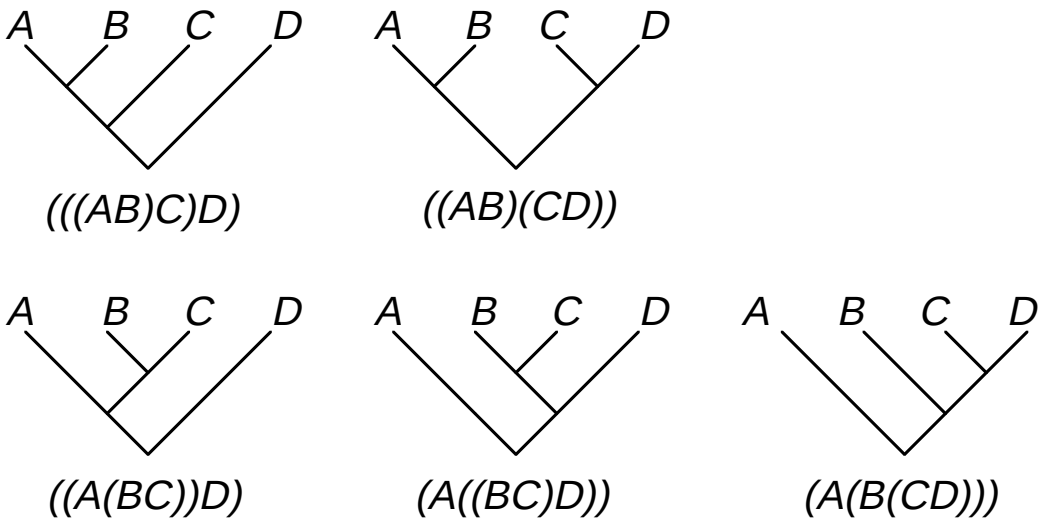


Figure 23: five non-ambiguous subassembly sequences of a four part assembly

can form through random mating. The first three results (a), (b) and (c) specify no fixed subassembly sequences. In other words, the parts can be assembled in *any* of the five subassembly sequences shown in Figure 23, *i.e.* the design encodes $\{ABCD\}$. A fixed subassembly sequence $((AB)(CD))$ emerged for (d) $\mathbf{q} = (0.2, 0.0, 0.2)$, which is realized by conformational changes of part B and part C after forming subassemblies AB and CD:

$$A + B \rightarrow AB' \tag{25}$$
$$C + D \rightarrow C'D \tag{26}$$
$$AB' + CD' \rightarrow AB'C'D \tag{27}$$

Figure 25 shows the results with $\mathbf{n}_0 = (10, 20, 10, 10)$. For (a) $\mathbf{q} = (0.0, 0.0, 0.0)$ and (b) $\mathbf{q} = (0.2, 0.0, 0.0)$, a conformational link in part B causes a B–C bond to be made only *after* an A–B bond forms. The final assembly $ABCD$, therefore, is built in the order either $(((AB)C)D)$ or $((AB)(CD))$, hence the design encodes the subassembly sequence $\{(AB)CD\}$. In the case of (c) $\mathbf{q} = (0.0, 0.2, 0.0)$, on the other hand, a conformational link in part B causes a B–C bond to be made *before* an A–B bond forms. Therefore, the design encodes the subassembly sequences $\underline{((A(BC))D)}$, $(A((BC)D))$ or $(A(B(CD)))$. We refer to the set of these three subassembly sequences as $\overline{\{(AB)CD\}}$, since they are the subassembly sequences that are *not* represented by $\{(AB)CD\}$ among the five possible non-ambiguous subassembly sequences in Figure 23. As in the case of $\mathbf{n}_0 = (10, 10, 10, 10)$, the resulting design specifies a fixed subassembly sequence $((AB)(CD))$ for (d) $\mathbf{q} = (0.2, 0.0, 0.2)$.

The sequence $((AB)(CD))$ also emerged for $\mathbf{n}_0 = (10, 20, 20, 10)$, with (a) $\mathbf{q} = (0.0, 0.0, 0.0)$, (b) $\mathbf{q} = (0.2, 0.0, 0.0)$ and (d) $\mathbf{q} = (0.2, 0.0, 0.2)$, as shown in Figure 26. The design with (c) $\mathbf{q} = (0.0, 0.2, 0.0)$ encodes the subassembly sequence $(A(B(CD)))$, which takes the following three-step reactions:

$$C + D \rightarrow C'D \tag{28}$$
$$B + C'D \rightarrow B'C'D \tag{29}$$
$$A + B'C'D \rightarrow AB'C'D \tag{30}$$

Note in the above results (of Figures 24(d), 25(d), 26(a), (b), (c) and (d)), that *exactly* two conformational links, one in part B and one in part C, are actually used to encode the two non-ambiguous subassembly sequences $((AB)(CD))$ and $(A(B(CD)))$. Other links are *non-functional* (do not cause conformational changes of a part) or *redundant* (cause conformational changes that do not affect assembly sequences). Similarly, as shown in Figures 25(a), (b) and (c), only one conformational link in part B is required to encode $\{(AB)CD\}$ and $\overline{\{(AB)CD\}}$, and no conformational link is required to encode $\{ABCD\}$ (see Figures 24(a), (b) and (c))[15].

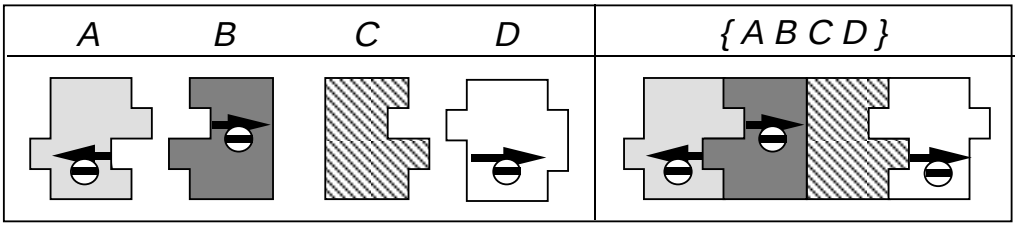
4.4 Rate equation analyses of four part randomized assembly

Rate equations (14) of four-part randomized assembly are formulated in a similar way to the three-part case in Section 4.2. The yield of the final assembly $ABCD$ is then compared for *all* the subassembly sequences which can be encoded by our conformational switch model described in Section 2.1. Such subassembly sequences can be enumerated by listing combinations of *functional* and *non-redundant* conformational links in the parts, *i.e.* by listing all combinations of conformational links which are necessary and sufficient to encode a set of non-ambiguous subassembly sequences. Since our desired assembly is $ABCD$, any conformational links in part A and part D are non-functional/redundant. Also, one of any two conformational links in a part pointing the same direction is non-functional/redundant since it is not possible to induce conformational changes via two such conformational links (see Section 2.1). Even when two conformational links in a part are pointing in the opposite directions, one of them is still non-functional/redundant since a bond cannot be destroyed once it is formed.

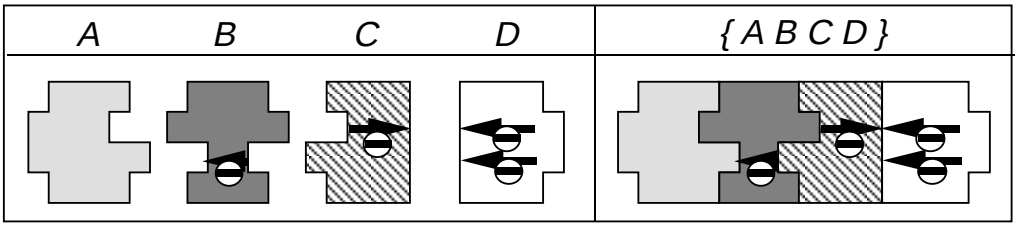
The above discussion leaves us only eight non-redundant combinations of conformational links, which encode five ambiguous subassembly sequences $\{ABCD\}$, $\{(AB)CD\}$, $\overline{\{(AB)CD\}}$, $\{AB(CD)\}$ and

[15] Conformational changes of part D in Figures 24(a) and (c) do not affect assembly sequences, therefore the corresponding links are redundant.

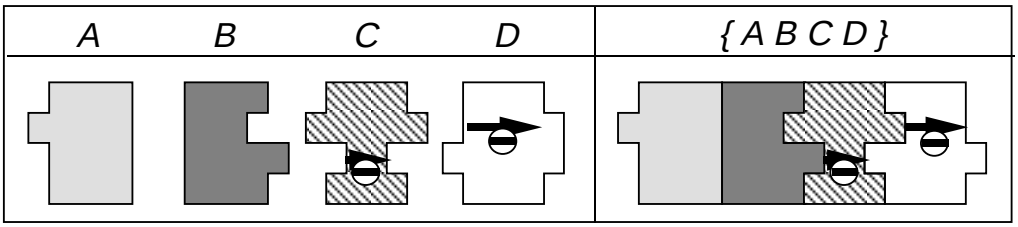
(a) ***q*** *= (0.0, 0.0, 0.0); fitness = 10.00; n = 32*



(b) ***q*** *= (0.2, 0.0, 0.0); fitness = 8.34; n = 2*



(c) ***q*** *= (0.0, 0.2, 0.0); fitness = 8.42; n = 2*



(d) ***q*** *= (0.2, 0.0, 0.2); fitness = 7.63; n = 3*

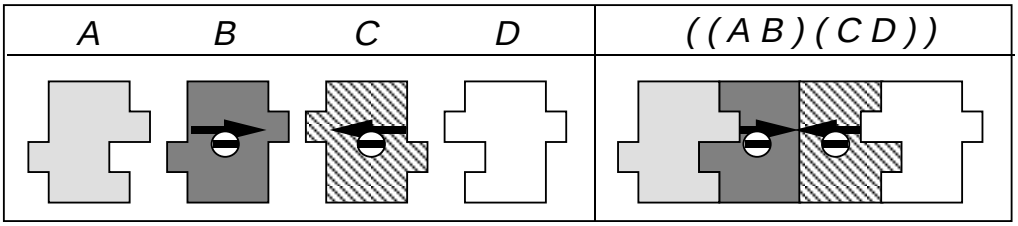
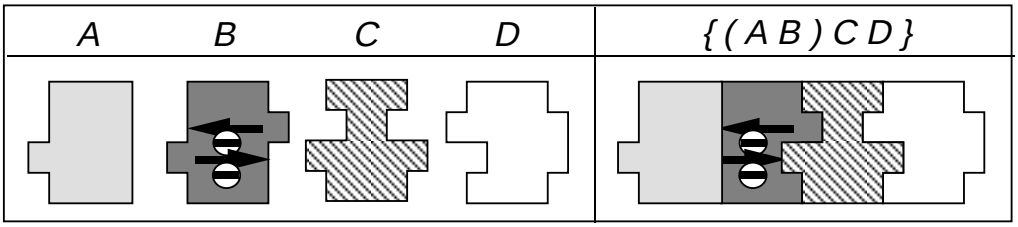
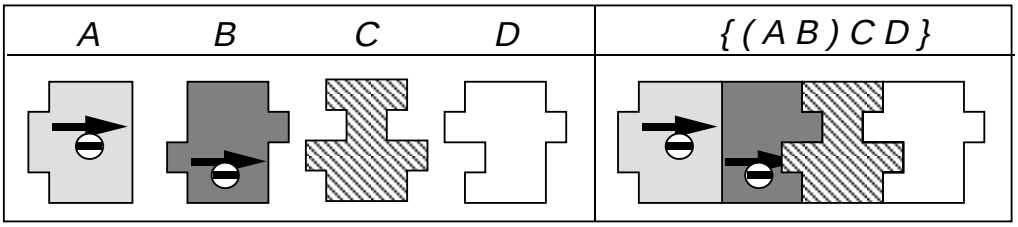


Figure 24: best designs with $\mathbf{n}_0 = (10, 10, 10, 10)$

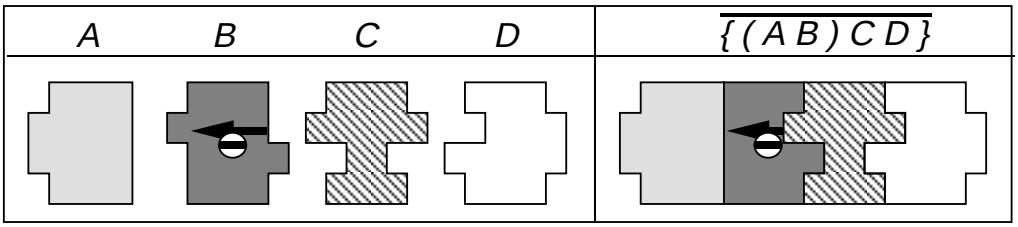
(a) ***q*** = (0.0, 0.0, 0.0); fitness = 9.84; n = 2



(b) ***q*** = (0.2, 0.0, 0.0); fitness = 8.16; n = 1



(c) ***q*** = (0.0, 0.2, 0.0); fitness = 8.28; n = 2



(d) ***q*** = (0.2, 0.0, 0.2); fitness = 8.32; n = 1

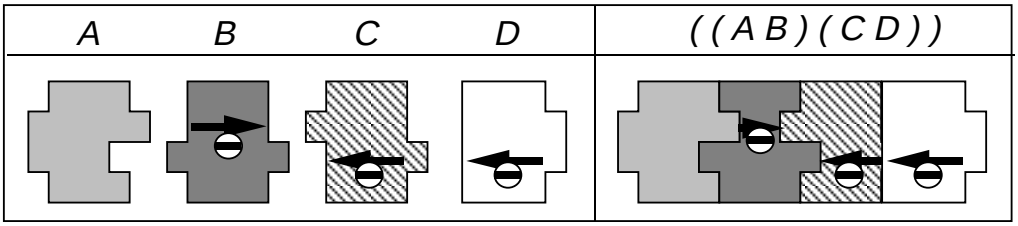
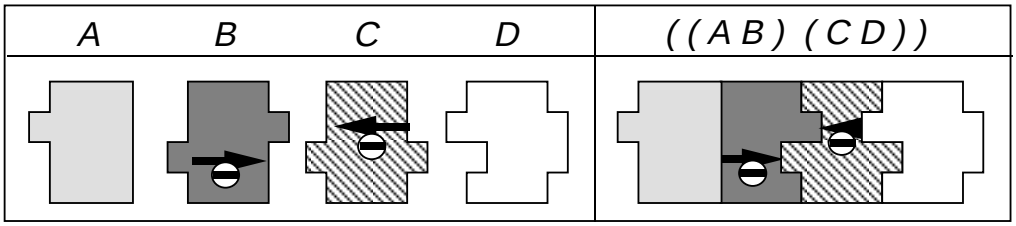
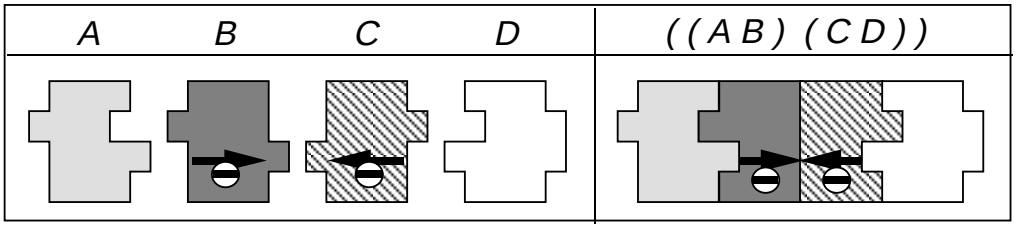


Figure 25: best designs with $\mathbf{n}_0 = (10, 20, 10, 10)$

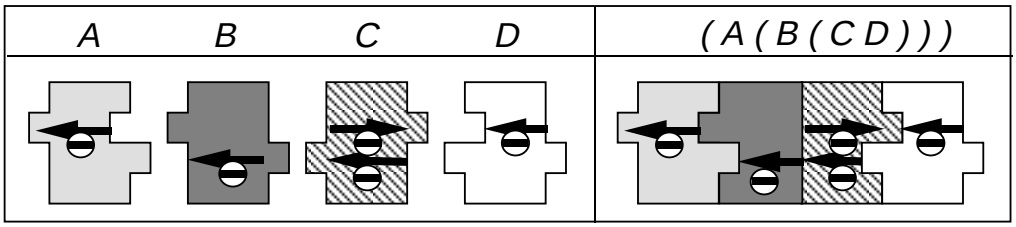
(a) ***q*** = (0.0, 0.0, 0.0); fitness = 9.48; n = 1



(b) ***q*** = (0.2, 0.0, 0.0); fitness = 8.08; n = 2



(c) ***q*** = (0.0, 0.2, 0.0); fitness = 7.98; n = 1



(d) ***q*** = (0.2, 0.0, 0.2); fitness = 7.24; n = 1

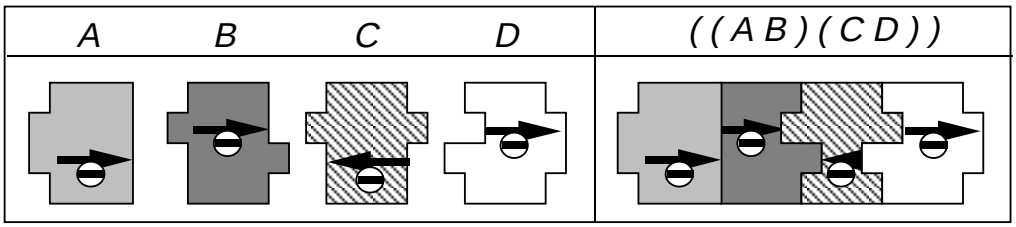


Figure 26: best designs with $\mathbf{n}_0 = (10, 20, 20, 10)$

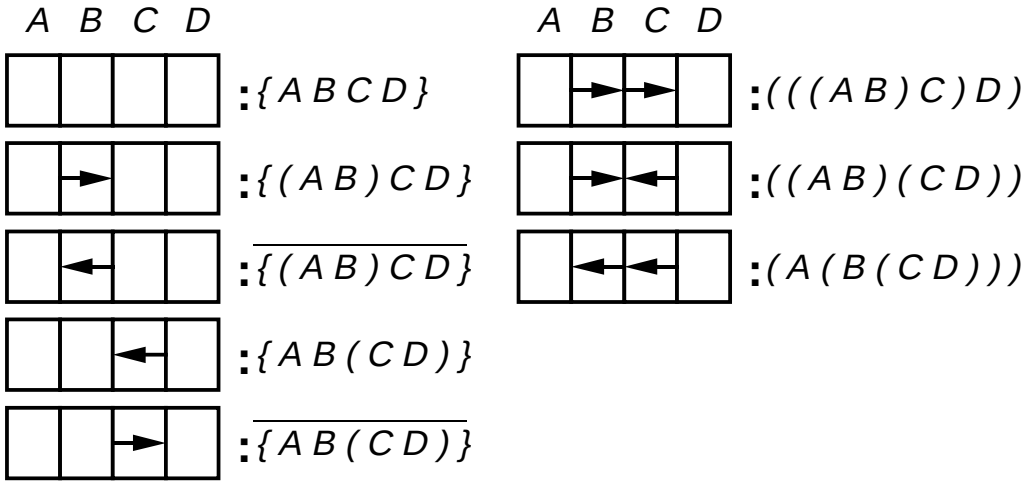


Figure 27: eight possible subassembly sequences

$\overline{\{AB(CD)\}}$, and three non-ambiguous subassembly sequences $(((AB)C)D)$, $((AB)(CD))$ and $(A(B(CD)))$, as illustrated in Figure 27[16]. It is shown, therefore, that our conformational switch model *cannot* encode two of the non-ambiguous subassembly sequences $((A(BC))D)$ and $(A((BC)D))$ in Figure 23. This is due to the fact that our conformational switch model does not allow propagation of conformational change through parts[17]. The rate equations for each of the eight subassembly sequences in Figure 28 are formulated and solved numerically to compare the yield of the final assembly $ABCD$. The results are obtained with three different initial conditions: $\mathbf{n}_0 = (10, 10, 10, 10)$, $\mathbf{n}_0 = (10, 20, 10, 10)$ and $\mathbf{n}_0 = (10, 20, 20, 10)$. For each of the three initial conditions, four different defect probabilities are tried: $\mathbf{q} = (0.0, 0.0, 0.0)$, $\mathbf{q} = (0.2, 0.0, 0.0)$, $\mathbf{q} = (0.0, 0.2, 0.0)$, $\mathbf{q} = (0.2, 0.0, 0.2)$. These $3 \times 4 = 12$ conditions correspond to the conditions of GA runs shown in Figures 24, 25 and 26. Figures 28–31 show the solution of the equation 14 with $\mathbf{n}_0 = (10, 10, 10, 10)$ and with $\mathbf{q} = (0.0, 0.0, 0.0)$, $\mathbf{q} = (0.2, 0.0, 0.0)$, $\mathbf{q} = (0.0, 0.2, 0.0)$, $\mathbf{q} = (0.2, 0.0, 0.2)$, respectively. For the first three cases, the sequence $\{ABCD\}$ scores the best at 1400 iterations, whereas it is outperformed by $((AB)(CD))$ after approximately 500 iterations for $\mathbf{q} = (0.2, 0.0, 0.2)$[18]. In all cases, the results are consistent with the ones by GA shown in Figure 24.

The solutions with $\mathbf{n}_0 = (10, 20, 10, 10)$ are shown in Figures 32–35. In this case, the yields of the sequences $\{ABCD\}$, $\{AB(CD)\}$ and $\overline{\{AB(CD)\}}$ drop to approximately 50% of the maximum possible yield. This situation is similar to the case of the sequence $\{ABC\}$ with $\mathbf{n}_0 = (10, 20, 10)$ in the three part assembly, where excess production of AB and BC at the early stage of iteration causes the shortage of A's and B's later on. The sequences which do not specify the assembly order of A, B and C (or CD) perform poorly due to excess production of intermediate subassemblies such as AB, BC or BCD. As a consequence, the sequences $\{(AB)CD\}$ and $\overline{\{(AB)CD\}}$ yield the best with $\mathbf{q} = (0.0, 0.0, 0.0)$ (Figure 32), the sequence $((AB)(CD))$ is the best with $\mathbf{q} = (0.2, 0.0, 0.0)$ (Figure 33) and $\mathbf{q} = (0.2, 0.0, 0.2)$ (Figure 35), and the sequence $\overline{\{(AB)CD\}}$ is the best with $\mathbf{q} = (0.0, 0.2, 0.0)$ (Figure 34). The GA found a design that encodes the optimal subassembly sequence for all cases except $\mathbf{q} = (0.2, 0.0, 0.0)$, where the design encodes the suboptimal sequence $\{(AB)CD\}$ within 5% of the yield of the optimal sequence (see Figure 25(b)). Similar drops of yield in some subassembly sequences are observed in the solutions with $\mathbf{n}_0 = (10, 20, 20, 10)$. As shown in Figures 36–39, however, the drop occurs to *all* of the ambiguous subassembly sequences, $\{ABCD\}$, $\{(AB)CD\}$, $\overline{\{(AB)CD\}}$, $\{AB(CD)\}$ and $\overline{\{AB(CD)\}}$. The sequence $((AB)(CD))$ yields the best for $\mathbf{q} = (0.0, 0.0, 0.0)$, $\mathbf{q} = (0.2, 0.0, 0.0)$, and $\mathbf{q} = (0.2, 0.0, 0.2)$, and the se-

[16]Some extensions are necessary to our conformational switch model, in order to encode $\{A(BC)D\}$. This issue is discussed in Section 5.1.

[17]An extension of the conformational switch model which can encode $((A(BC))D)$ and $(A((BC)D))$ is discussed in Section 5.1.

[18]Recall the robot bin-picking simulation used in the GA search terminates at 1400 iterations.

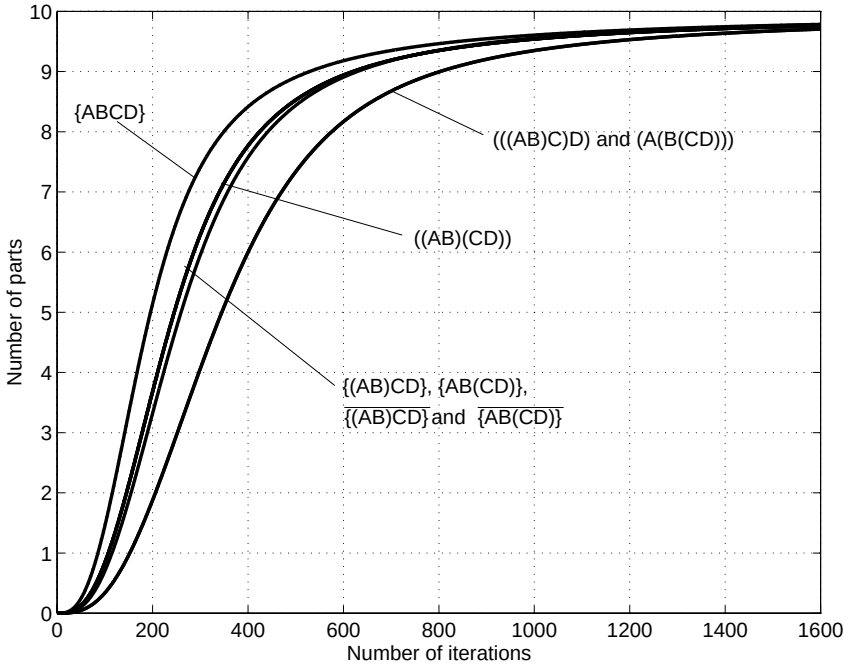


Figure 28: solution with $\mathbf{n}_0 = (10, 10, 10, 10)$ and $\mathbf{q} = (0.0, 0.0, 0.0)$

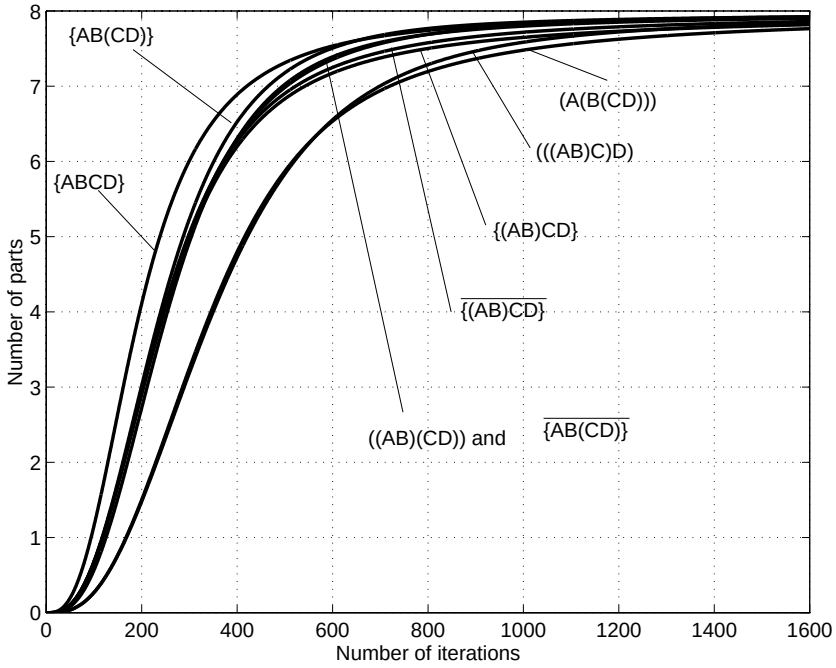


Figure 29: solution with $\mathbf{n}_0 = (10, 10, 10, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.0)$

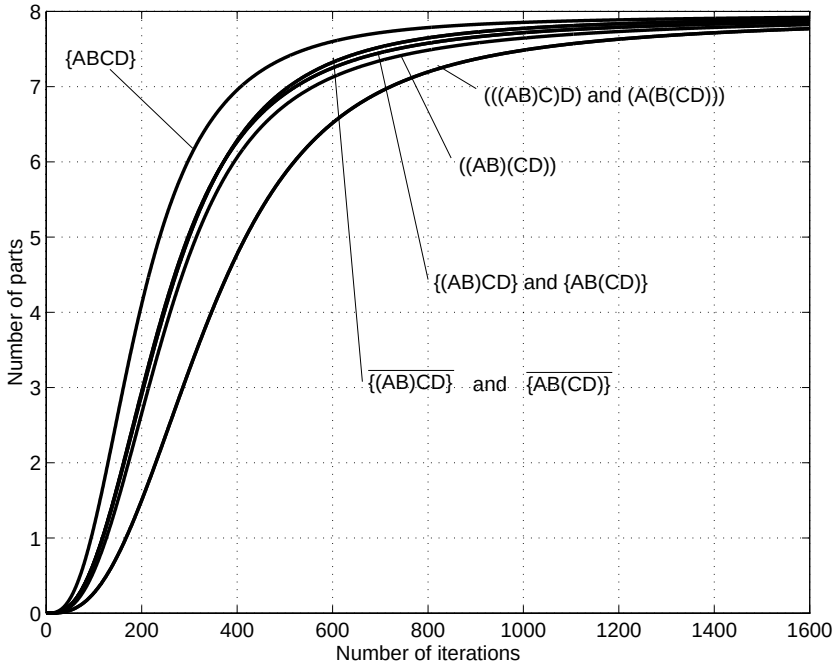


Figure 30: solution with $\mathbf{n}_0 = (10, 10, 10, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$

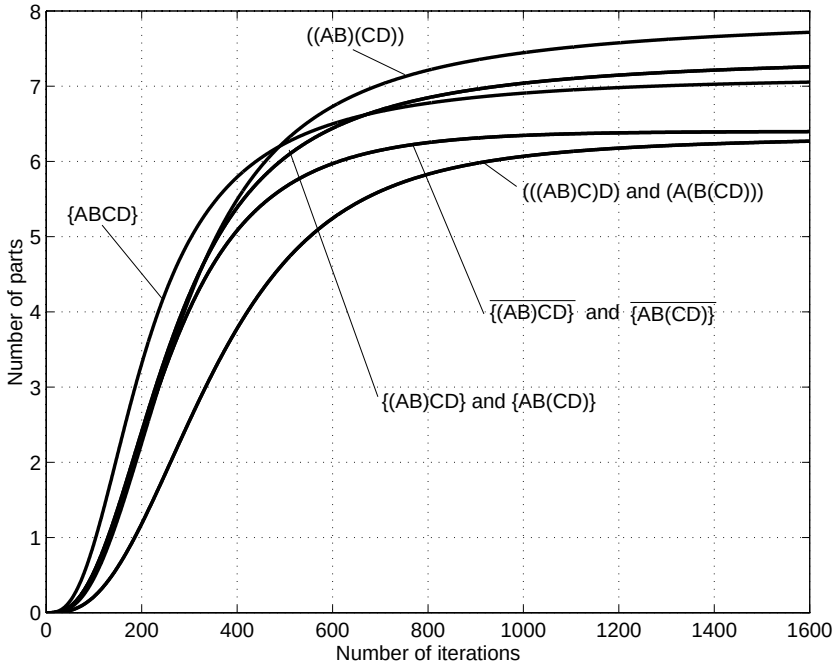


Figure 31: solution with $\mathbf{n}_0 = (10, 10, 10, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.2)$

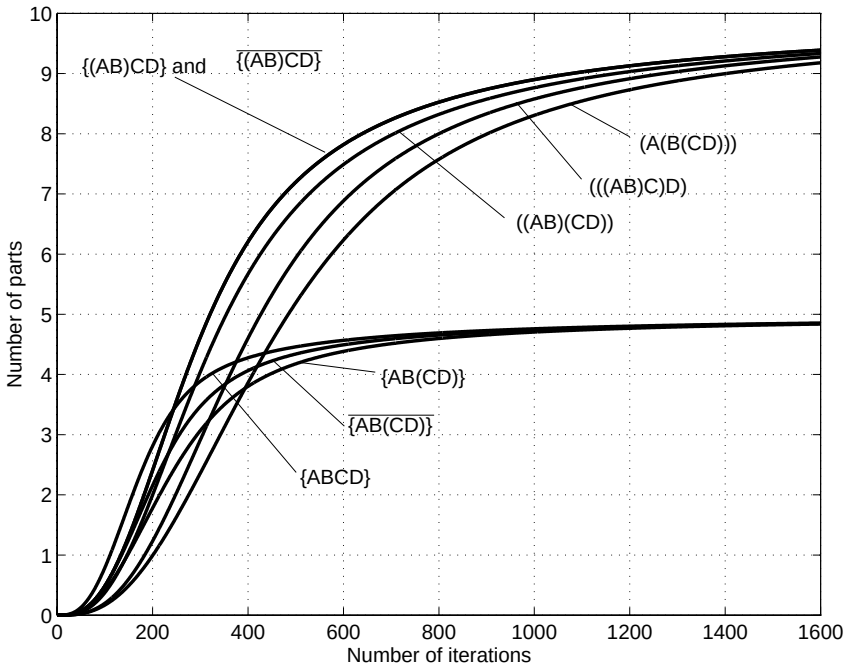


Figure 32: solution with $\mathbf{n}_0 = (10, 20, 10, 10)$ and $\mathbf{q} = (0.0, 0.0, 0.0)$

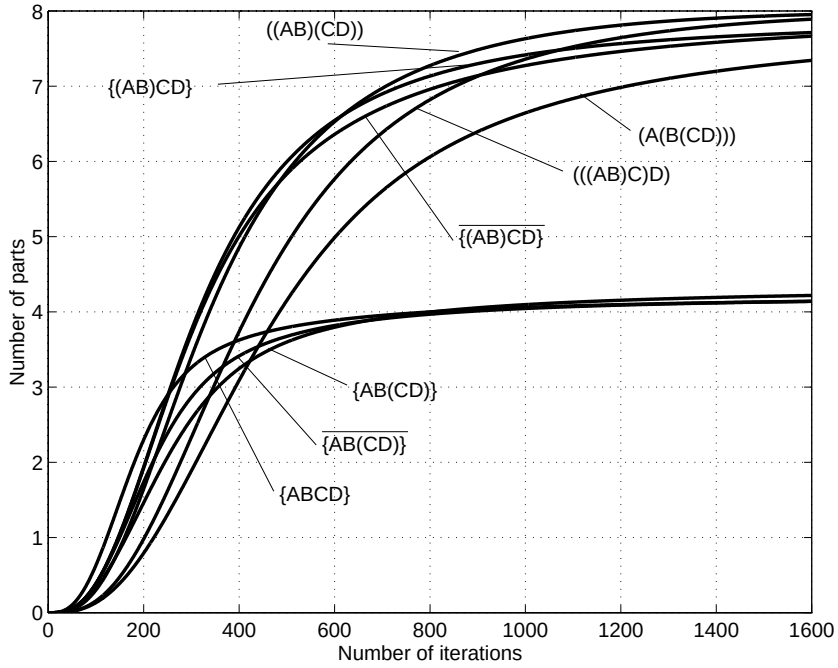


Figure 33: solution with $\mathbf{n}_0 = (10, 20, 10, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.0)$

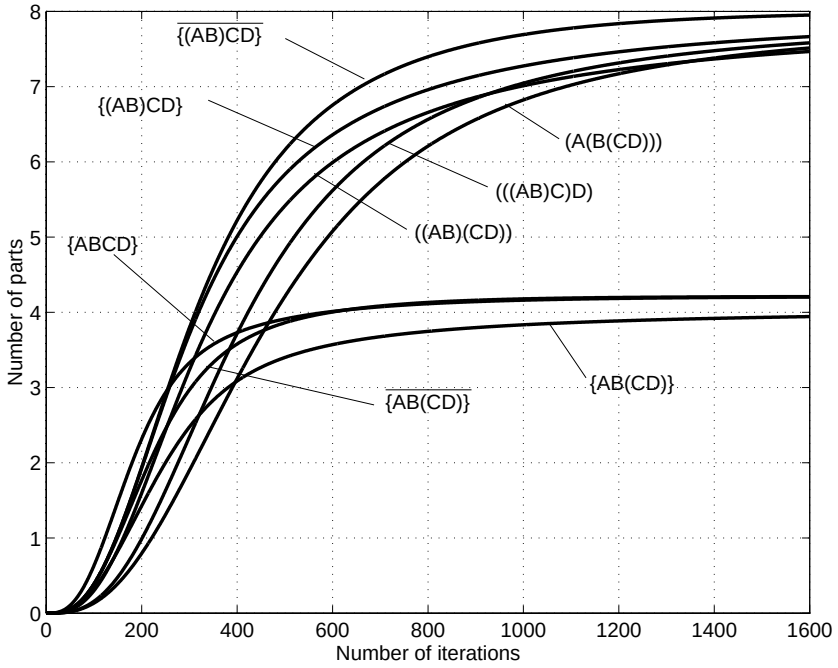


Figure 34: solution with $\mathbf{n}_0 = (10, 20, 10, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$

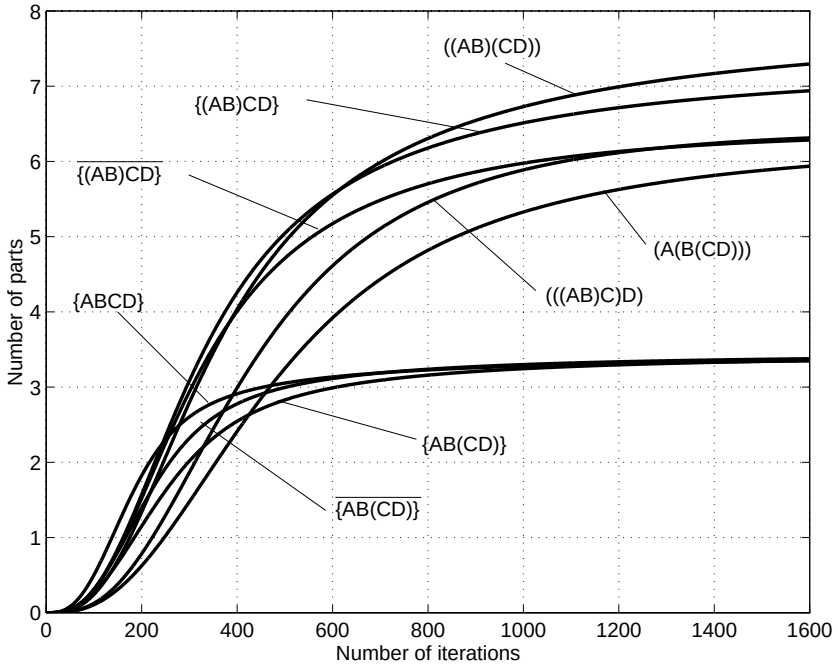


Figure 35: solution with $\mathbf{n}_0 = (10, 20, 10, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.2)$

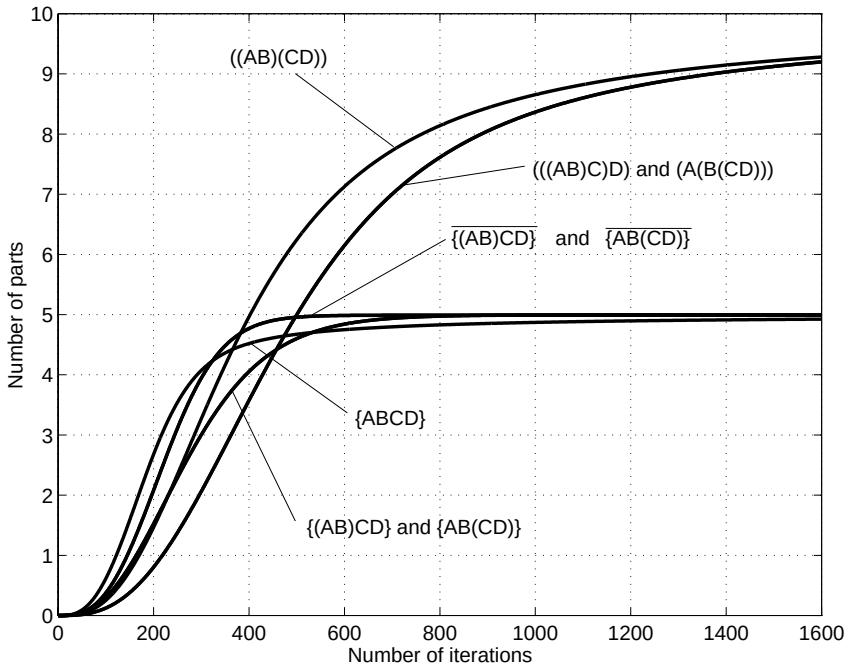


Figure 36: solution with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.0, 0.0, 0.0)$

quences $(((AB)C)D)$ and $(A(B(CD)))$ yield the best for $\mathbf{q} = (0.0, 0.2, 0.0)$. These results are consistent with the ones found by the GA shown in Figure 26.

5 Discussion

As indicated in our examples in the previous sections, a particular subassembly sequence is generated due to conformational changes of parts during one-dimensional randomized assembly. An optimal subassembly sequence that maximizes the yield of a desired assembly can be found via genetic search over a space of parameterized conformational switch designs, rather than a space of subassembly sequences. The resulting switch design encodes the optimal subassembly sequence so that the desired assemblies are put together only in that sequence. The results of genetic search and rate equation analyses reveal that the optimal subassembly sequence depends on the initial concentration of parts and the defect probabilities during randomized assembly. More specifically, the results seem to indicate the following "rules of thumb" to design conformational switches for the general n-part one-dimensional assembly:

- Abundant parts should be assembled earlier rather than later.
- Parts with high defect probability should be assembled earlier rather than later.

There are several issues to note when interpreting the above results: (1) encoding power of a conformational switch model, (2) switch design vs. sequence design, and (3) the sensitivity of yield to the initial concentration of parts.

5.1 Encoding power of a conformational switch model

One must note the encoding power of a conformational switch model, in order to say that the switch design obtained by a GA actually encodes the optimal subassembly sequence. There could be a subassembly sequence that *cannot* be encoded by a conformational switch design, but that *is* better than

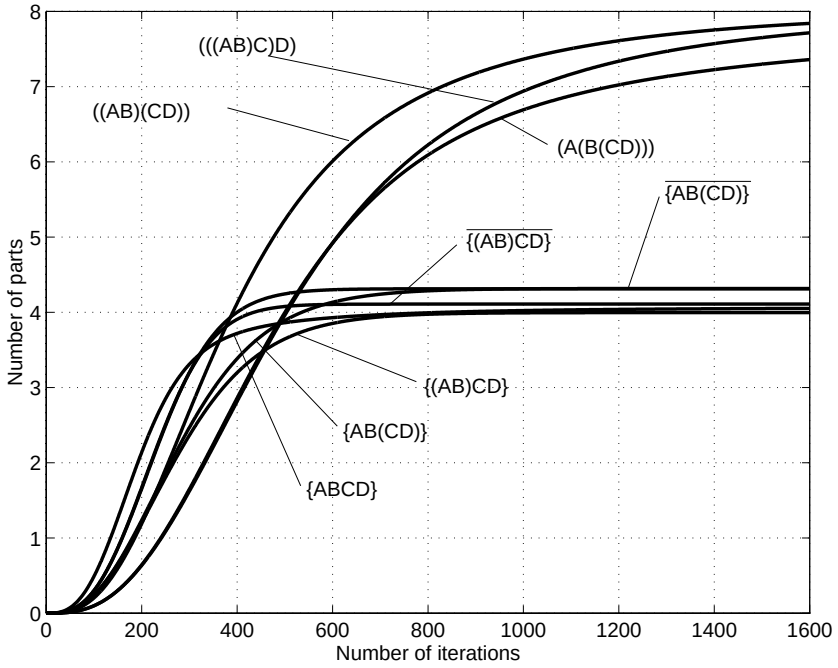


Figure 37: solution with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.0)$

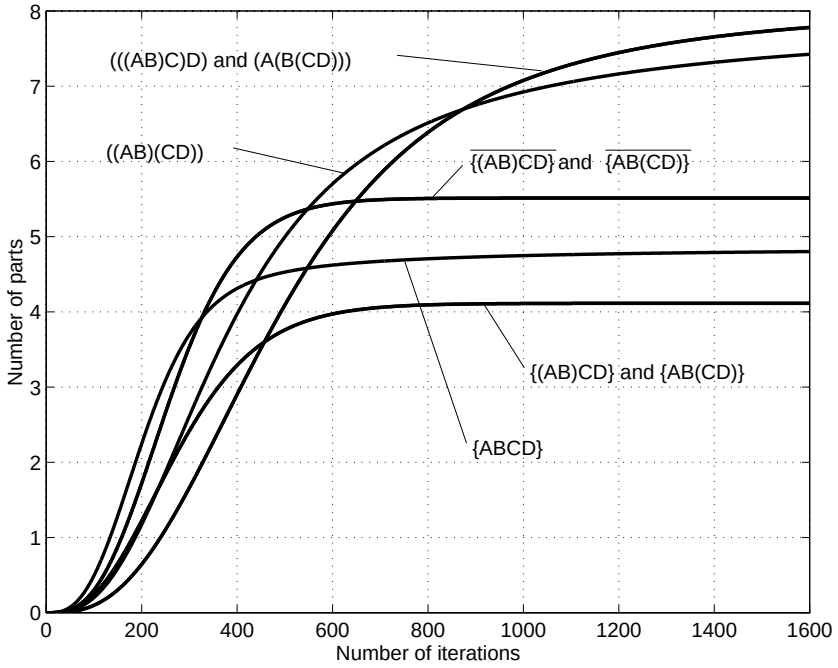


Figure 38: solution with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$

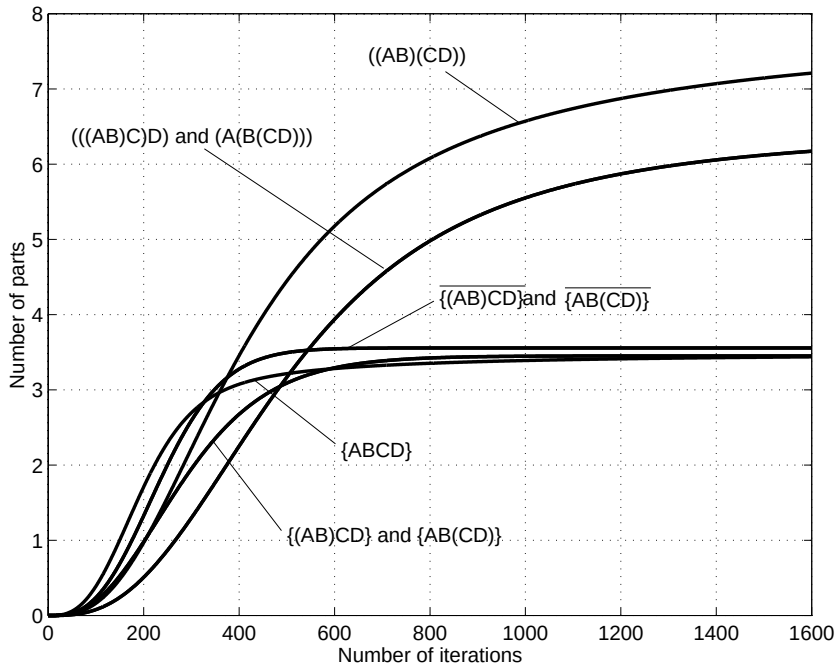


Figure 39: solution with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.2)$

any subassembly sequences encoded by the switch design. In fact, $((A(BC))D)$ and $(A((BC)D))$, the subassembly sequences our conformational switch model *cannot* encode, yield better than $(A(B(CD)))$, the best sequence obtained by the GA in the four part randomized assembly with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$, as shown in Figure 40. It should be emphasized, however, that the sequence found by the GA is the best among the sequences that can be encoded by our conformational switch design.

For comparison, Figure 40 also shows the solution of rate equations for three other un-encodable subassembly sequences, $(\{ABC\}D)$, $(A\{BCD\})$ and $\{A(BC)D\}$. The sequences $(\{ABC\}D)$ and $(A\{BCD\})$ actually outperform the best encodable sequences $(((AB)C)D)$ and $(A(B(CD)))$. The sequences $((A(BC))D)$ and $(A((BC)D))$ also perform better than any encodable sequences in the cases with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.0, 0.0, 0.0)$, and with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.2, 0.0, 0.0)$. The differences in yield, however, are marginal.

Only a few modifications/extensions are necessary in the current conformational switch model to encode the five un-encodable subassembly sequences in Figure 40, $\{A(BC)D\}$, $((A(BC))D)$, $(A((BC)D))$, $(\{ABC\}D)$ and $(A\{BCD\})$. In order to encode $\{A(BC)D\}$, we only need to modify priority to upstream propagation (Figure 6) such that conformational changes can propagate in *both* direction when it is possible[19]. Figure 41 illustrates an example of a conformational switch design that encodes $\{A(BC)D\}$ with this modification.

The sequences $((A(BC))D)$, $(A((BC)D))$, $(\{ABC\}D)$ and $(A\{BCD\})$ can be encoded by introducing the *sliding bar mechanism* described in [16], which allows propagation of conformational change through multiple parts, and an additional "digit"[20]. The definition of three types of bonding can be defined, for example, as analogous to the two-digit case. Namely, two bond sites form a stable bond if $\forall i \ a_i + b_i \leq 0$, unstable bond if $\exists_1 j \ a_j + b_j = 1$ and $\forall i \neq j \ a_i + b_i \leq 0$, and no bond otherwise, where $i, j \in \{1, 2, 3\}$. Figure 42 illustrates an example of a conformational switch design that encodes

[19]Appropriate changes in the definition of unstable bond are also required.

[20]It seems that there is no two-digit switch designs that encodes these four sequences. Proving/disproving this would be a part of future work.

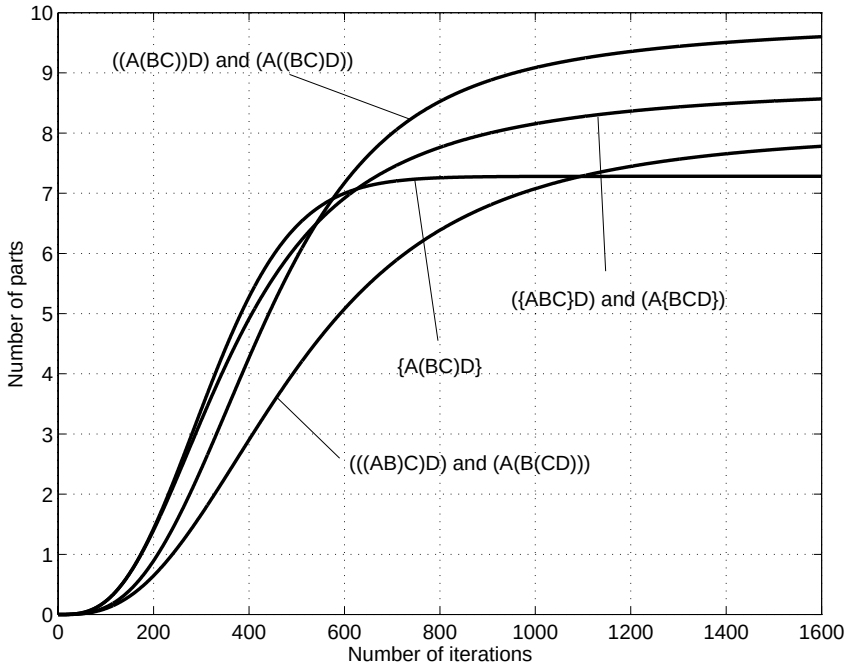


Figure 40: solution with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$: comparison with un-encodable subassembly sequences

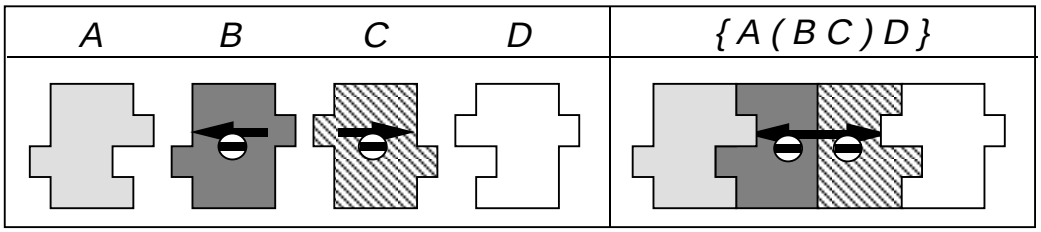


Figure 41: a conformational switch design that encodes $\{A(BC)D\}$

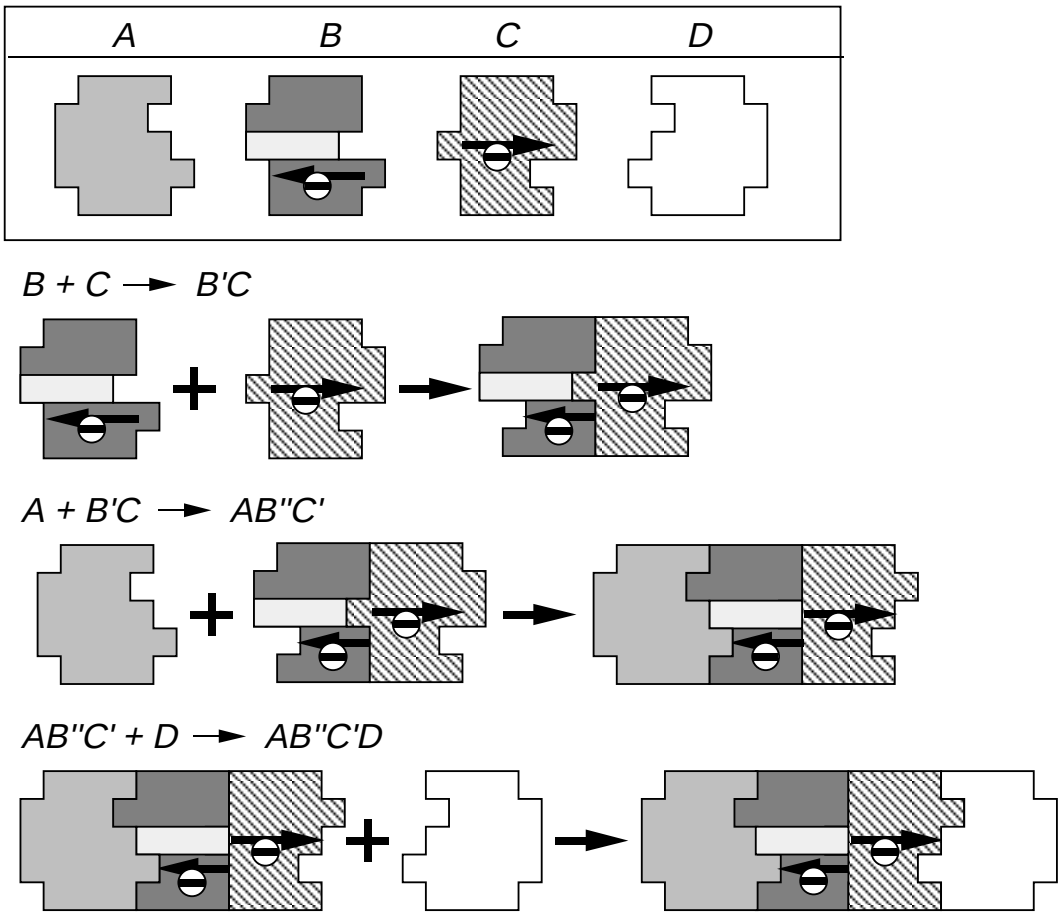


Figure 42: a conformational switch design that encodes $((A(BC))D)$

$((A(BC))D)$ with this extension and its three-step reactions. Note that the minus link in part B allows parts A and B to bond only *after* parts B and C bond. By removing the link, therefore, one can construct a switch design that encodes $(\{ABC\}D)$, as shown in Figure 43. Conformational switch designs that encode $(A((BC)D))$ and $(A\{BCD\})$ can be obtained by horizontally flipping the designs in Figures 42 and 43, respectively.

5.2 Switch design vs. sequence design

In this paper, the space of parameterized conformational switch designs is searched to find the switch designs that maximize the yield of a desired assembly during one-dimensional randomized assembly. And then, the resulting designs are analyzed using the rate equation to confirm that the subassembly sequence encoded by the switch designs is in fact the optimal. The size of the search space is 2^{mn} where m is the number of bits used to encode the parameters for a part design, and n is the number of parts in the desired assembly. This grows exponentially as n increases. An alternative approach is to search over the space of subassembly sequences to find the optimal sequence, and then find the conformational switch designs that encode the subassembly sequence. In this case, the size of search space is $\Omega(2^n)$, where Ω denotes an asymptotic lower bound [21].

Even though the complexity of the problems are exponential in the both approaches, the first has

[21] The proof of this fact is found in Appendix C

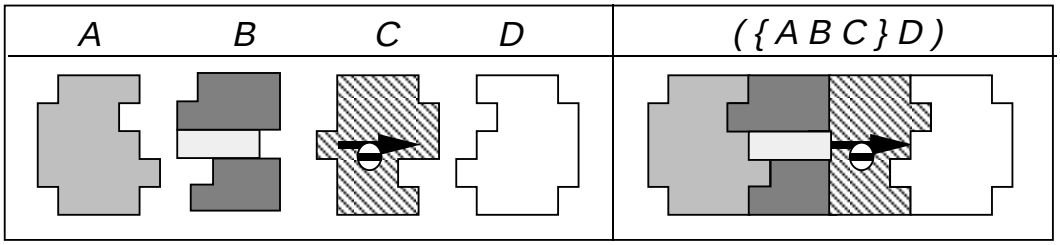


Figure 43: a conformational switch design that encodes $(\{ABC\}D)$

several advantages over the second. First, by searching the space of parameterized switch designs directly, we find *only* the subassembly sequences which can be encoded by the given conformational switch model. Since some subassembly sequences cannot be realized by a conformational switch model, we must add additional constraints when searching over the space of subassembly sequences. It is difficult in general, however, to know exactly which subassembly sequences can be encoded by a given conformational switch model. Also, in the second approach, optimal switch designs corresponding to the optimal subassembly sequences must be generated, whereas in the first approach the optimal switch designs are found directly as a result of the search. The first approach also seems to fit naturally within the framework of genetic algorithms since there is a direct analogy between the short-order building blocks in the binary representation and the complementary bonding sites in the geometric representation of the conformational switches.

5.3 Sensitivity of yield to the initial concentration of parts

The optimal subassembly sequences encoded by conformational switch designs are $\{ABCD\}$ for $\mathbf{n}_0 = (10, 10, 10, 10)$ with no defect, *i.e.* no order of assembly is specified as shown in Figure 28. On the other hand, $\{ABCD\}$ yields are very low with different initial concentration of parts as shown in Figures 32 and 36. This seems to be a general property of ambiguous subassembly sequences: their yield change dramatically depending on the initial part concentration. In fact, the yield of ambiguous subassembly sequences are quite sensitive to the change in the initial concentration of parts. Figure 44 shows the solution of rate equations with $\mathbf{n}_0 = (10, 11, 11, 10)$ and $\mathbf{q} = (0.0, 0.0, 0.0)$ (compare with Figure 28). Even with the slightest change in $\mathbf{n}_0$, the yield of ambiguous subassembly sequences are approximately 10% lower than the ones of non-ambiguous subassembly sequences. On the other hand, the initial part concentration has little effect on the yield of non-ambiguous subassembly sequences. This robustness of non-ambiguous subassembly sequences against the initial part concentration is a great advantage in real-world randomized assembly processes, *e.g.* biological self-assembly, where realizing a precise and uniform concentration of parts is extremely difficult.

Acknowledgements

This work is supported by the National Science Foundation with a Presidential Young Investigator's grant (DDM-9058415). Matchable funds for this grant have been provided by Schlumberger Inc.. These sources of support are gratefully acknowledged. Additionally, the work was carried out using the computational facilities of the Computer-Aided Design Laboratory at the Massachusetts Institute of Technology, Department of Mechanical Engineering. This support is also gratefully acknowledged.

References

[1] B. Berger, P. W. Shor, L. Tucker-Kellog, and J. King. Local rule-based theory of virus shell assembly. In *Proceedings of the National Academy of Science, USA*, pages 7732–7736, 1994. Vol. 91.

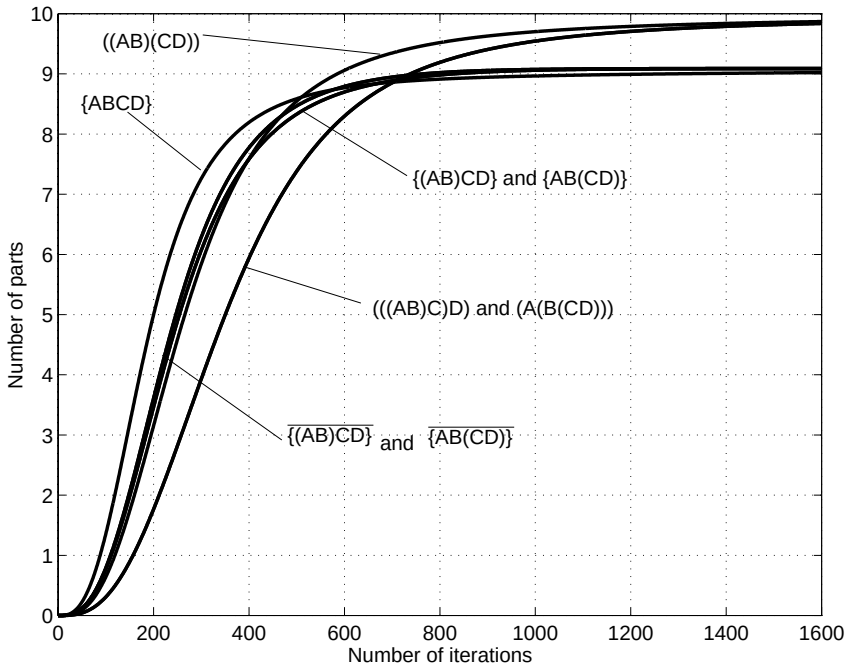


Figure 44: solution with $\mathbf{n}_0 = (10, 11, 11, 10)$ and $\mathbf{q} = (0.0, 0.0, 0.0)$

[2] S. Casjens. and J. King. Virus assembly. *Annual Review of Biochemistry*, 44:555–604, 1975.

[3] M. B. Cohn, C.-J. Kim, and A. P. Pisano. Self-assembling electrical networks: an application of micromachining technology. In *Transducers '91: 1991 Sixth International Conference on Solid-State Sensors and Actuators*, pages 490–493, New York, New York, 1991. IEEE.

[4] H. R. Crane. Principles and problems of biological growth. *The Scientific Monthly*, 70:376–389, 1950.

[5] R. A. Crowther, E. V. Lenk, Y. Kikuchi, and J. King. Molecular reorganization in the hexagon to star transition of the baseplate of bacteriophage T4. *Journal of Molecular Biology*, 116:489–523, 1977.

[6] N. S. Goel and R. L. Thompson. Movable finite automata (MFA): A new tool for computer modeling of living systems. In C. G. Langton, editor, *Artificial Life: the Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, pages 317–340, Los Alamos, New Mexico, September 1987. Addison Wesley.

[7] N. S. Goel and R. L. Thompson. *Computer Simulations of Self-organization in Biological Systems.* Croom Helm, London, England, 1988.

[8] N. S. Goel and R. L. Thompson. Movable finite automata (MFA) models for biological systems II: Protein biosynthesis. *Journal of Theoretical Biolology*, 134:9–49, 1988.

[9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[10] K. Hosokawa, I. Shimoyama, and H. Miura. Dynamics of self-assembling systems: Analogy with chemical kinetics. *Artificial Life*, 1(4):413–427, 1994.

[11] P. H. Moncevicz. Orientation and insertion of randomly presented parts using vibratory agitation. Master's thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, 1991.

[12] P. H. Moncevicz and M. J. Jakiela. Method and appratus for automatic parts assembly. United States Patent 5,155,895, October 20 1992.

[13] P. H. Moncevicz, M. J. Jakiela, and K. T. Ulrich. Orientation and insertion of randomly presented parts using vibratory agitation. In A. H. Soni, editor, *Proceedings of the ASME 3rd Conference on Flexible Assembly Systems*, pages 41–47, New York, NY, September 1991. The American Society of Mechanical Engineers. DE-Vol. 33.

[14] L. S. Penrose. Self-reproducing machines. *Scientific American*, 200:105–114, June 1959.

[15] R. Rosen. Subunit and subassembly process. *Journal of Molecular Biology*, 28:415–422, 1970.

[16] K. Saitou and M. J. Jakiela. Automated optimal design of mechanical conformational switches. *Artificial Life*, 2(2):129–156, 1995.

[17] R. L. Thompson and N. S. Goel. A simulation of T4 bacteriophage assembly and operation. *BioSystems*, 18:23–45, 1985.

[18] R. L. Thompson and N. S. Goel. Movable finite automata (MFA) models for biological systems I: Bacteriophage assembly and operation. *Journal of Theoretical Biology*, 131:351–385, 1988.

[19] J. D. Watson, N. H. Hopkins, J. W. Roberts, J. A. Steitz, and A. M. Weiner. *Molecular Biology of the Gene*. Benjamin/Cummings, Menlo Park, California, 1987.

[20] H. J. Yeh and J. S. Smith. Fluidic self-assembly of GaAs microstructures on Si substrates. *Sensors and Materials*, 6(6):319–332, 1994.

A Rosen's subassembly model

This appendix outlines the derivation of Rosen's subassembly model, equations (1) and (2), described in Section 1.3.4.

Let N be the number of subassembly stages, and r_i be the number of subassemblies[22] produced at the $(i-1)$-th assembly stage, which are incorporated into a single subassembly produced at the i-th assembly stage. Hence the number of elementary units in the final assembly L is:

$$L = r_1 \cdot r_2 \cdot \ldots \cdot r_N \tag{31}$$

We assume Crane's assumptions hold: 1) with probability q two subassemblies are put together wrongly, causing the resulting subassembly to be defective, and 2) the defective subassemblies cannot be incorporated into the subsequent subassemblies, so the elementary units in the defective subassemblies are completely wasted. These assumptions give the following recurrent expression of ν_i, the number of non-defective subassemblies produced at the i-th assembly stage:

$$\begin{cases} \nu_i &= (1-q)^{r_i} \left(\dfrac{\nu_{i-1}}{r_i} \right); \qquad i \in \{1, 2, \ldots, N\} \\ \nu_0 &= M \end{cases} \tag{32}$$

where M is the number of elementary units in the initial pool. The above recurrence is easily solved and yields ν_N, the number of non-defective final assembly:

$$\nu_N = (1-q)^{r_1 + r_2 + \ldots + r_N} \left(\frac{M}{L} \right) \tag{33}$$

[22] Note that r_1 is the number of elementary units put together at the first assembly stage.

Using the equation (33), we can calculate the yields of Crane's two subassembly processes described in Section 1.3.4. The yield of the first, three-stage subassembly process:

$$\nu_N = (1 - 0.01)^{10+10+10} \left(\frac{1,000,000}{1,000} \right) = 739 \tag{34}$$

and for the second, one-stage subassembly process:

$$\nu_N = (1 - 0.01)^{1,000} \left(\frac{1,000,000}{1,000} \right) = 0.0432 \tag{35}$$

Our objective is to choose the $N + 1$ non-negative integers, $N, r_1, r_2, \ldots, r_N$, such that the expression (33) is maximized, subject to the non-linear constraint (31). Equivalently in integer programming formulation(equations (1) and (2) in Section 1.3.4):

$$\begin{aligned} \text{maximize} \quad & (1-q)^{r_1+r_2+\ldots+r_N} \left(\frac{M}{L} \right) \\ \text{subject to} \quad & L = r_1 \cdot r_2 \cdot \ldots \cdot r_N \\ & N \geq 0; \quad N \in \mathbf{Z} \\ & r_i \geq 0; \quad r_i \in \mathbf{Z}; \quad i \in \{1, 2, \ldots, N\} \end{aligned}$$

By assuming different q's at each subassembly stage, $q_1, q_2, \ldots, q_N$, the solution of the recurrence (32) becomes:

$$\nu_N = (1 - q_1)^{r_1}(1 - q_2)^{r_2} \ldots (1 - q_N)^{r_N} \left(\frac{M}{L} \right) \tag{36}$$

and the corresponding integer programming formulation (equations (5) and (6) in Section 1.3.4) becomes:

$$\begin{aligned} \text{maximize} \quad & (1-q_1)^{r_1}(1-q_2)^{r_2} \ldots (1-q_N)^{r_N} \left(\frac{M}{L} \right) \\ \text{subject to} \quad & L = r_1 \cdot r_2 \cdot \ldots \cdot r_N \\ & N \geq 0; \quad N \in \mathbf{Z} \\ & r_i \geq 0; \quad r_i \in \mathbf{Z}; \quad i \in \{1, 2, \ldots, N\} \end{aligned}$$

which in general cannot be solved in closed form.

B Proof of The Unique Factorization Theorem

Theorem 1 (Unique Factorization Theorem) *Let L be an positive integer, and $L = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$ be the unique factorization of L by the prime factors $p_1, p_2, \ldots, p_k$. For any factorization of $L = r_1 \cdot r_2 \cdot \ldots \cdot r_m$, the following inequality holds:*

$$\alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_k p_k \leq r_1 + r_2 + \ldots + r_m$$

Proof: Let l be a integer such that $0 \leq l \leq k$, and $N = \{1, 2, \ldots, k\}$. Without loss of generality, an arbitrary factorization of L can be written as

$$L = p_1^{\alpha_1 - u_1} \cdot p_2^{\alpha_2 - u_2} \cdot \ldots \cdot p_k^{\alpha_k - u_k} \cdot q_1 \cdot q_2 \cdot \ldots \cdot q_l \tag{37}$$

where for $i \in N$, $0 \leq u_i \leq \alpha_i$, and for $j \in \{1, 2, \ldots, l\}$,

$$q_j = \prod_{i \in N_j} p_i^{u_i} \tag{38}$$

and

$$N = \bigcup_{i=1}^{l} N_i; \qquad N_i \cap N_j = \begin{cases} \emptyset & \text{if } i \neq j \\ N_i & \text{if } i = j \end{cases} \tag{39}$$

Let S be the sum of the factors of this factorization, and S_0 be the sum of the factors of the unique prime factorization. We wish to show that $S_0 \leq S$ or equivalently $S - S_0 \geq 0$. Using the above notation, S can be written as

$$\begin{aligned} S &= (\alpha_1 - u_1)p_1 + (\alpha_2 - u_2)p_2 + \ldots + (\alpha_k - u_k)p_k + q_1 + q_2 + \ldots + q_l \\ &= (\alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_k p_k) - (u_1 p_1 + u_2 p_2 + \ldots + u_k p_k) + q_1 + q_2 + \ldots + q_l \end{aligned} \tag{40}$$

Since $S_0 = \alpha_1 p_1 + \alpha_2 p_2 + \ldots + \alpha_k p_k$,

$$\begin{aligned} S - S_0 &= q_1 + q_2 + \ldots + q_l - (u_1 p_1 + u_2 p_2 + \ldots + u_k p_k) \\ &= \gamma_1 + \gamma_2 + \ldots + \gamma_l \end{aligned} \tag{41}$$

where for $j \in \{1, 2, \ldots, l\}$

$$\gamma_j = q_j - \sum_{i \in N_j} u_i p_i \tag{42}$$

By Equation 38,

$$\begin{aligned} \gamma_j &= q_j - \sum_{i \in N_j} u_i p_i \\ &= \prod_{i \in N_j} p_i^{u_i} - \sum_{i \in N_j} u_i p_i \\ &= \left(\prod_{i \in N_j} p_i^{n_j u_i} \right)^{\frac{1}{n_j}} - \frac{\sum_{i \in N_j} n_j u_i p_i}{n_j} \end{aligned} \tag{43}$$

where $n_j = |N_j|$. We know for each $j \in \{1, 2, \ldots, l\}$, $\gamma_j \geq 0$ since (geometric mean)$\geq$(arithmetic mean). By Equation 41, therefore, $S - S_0 \geq 0$. ∎

C The size of the space of subassembly sequences

Theorem 2 *The number of non-ambiguous subassembly sequences S_n of an n-part one-dimensional assembly is $S_n = \Omega(2^n)$.*

Proof: Let k and l be positive integers such that $k+l = n$. Let us assume assembling n parts by assembling k parts and l parts separately, and then combining the k-part subassembly and l-part subassembly. In this particular case, the total number of non-ambiguous subassembly sequences of an n-part assembly is $S_k \cdot S_l$. Since in the general case k and l can be any positive integer such that $k + l = n$, S_n is the sum of all such cases:

$$\begin{aligned} S_n &= S_1 \cdot S_{n-1} + S_2 \cdot S_{n-2} + \ldots + S_{n-1} \cdot S_1 \\ &= \sum_{i=1}^{n-1} S_i S_{n-i} \end{aligned} \tag{44}$$

where $S_1 = 1$. Let

$$T_n = \sum_{i=1}^{n-1} T_i \tag{45}$$

and $T_1 = 1$. Since $\forall i \in \{1, 2, \ldots, n\}$, $S_i \geq 1$, for any positive integer n,

$$S_n \geq T_n \tag{46}$$

We know $T_n = \Theta(2^n)$ since

$$\begin{aligned} T_{n-1} + T_{n-2} + \ldots + T_1 &= (T_{n-2} + T_{n-3} + \ldots + T_1) + T_{n-2} + \ldots + T_1 \\ &= 2(T_{n-2} + T_{n-3} + \ldots + T_1) \\ &= 2 \cdot 2(T_{n-3} + \ldots + T_1) \\ &\vdots \\ &= 2^{n-2} T_1 = 2^{n-2} \end{aligned}$$

where Θ denotes an asymptotic tight bound. Equation (46) implies, therefore, $S_n = \Omega(2^n)$. ∎