

Metaheuristic Optimization in Certain and Uncertain Environments

2.1 Introduction

Until now, a variety of optimization methods have been used as effective tools for making a rational decision in manufacturing systems and will surely continue to do so. By virtue of the outstanding progress in computers, many applications have been carried out in the real world using commercial software that has been developed greatly. To understand the proper usage of software and the adequate choice of optimization method through revealing merits and demerits compared with recent metaheuristic approaches, it is essential for every practitioner to have basic knowledge of these methods.

We can always systematically define every optimization problem by the triplet of arguments $(x, f(x), X)$ where x is an n -dimensional vector called decision variable and $f(x)$ an objective function. Moreover, X denotes a subset of \mathbb{R}^n called an admissible region or a feasible region that is prescribed generally by a set of equality and/or inequality equations called constraints. Using these arguments, the optimization problem can be described generally and simply as follows:

$$[Problem] \quad \min f(x) \quad \text{subject to } x \in X.$$

The maximization problem can be handled in the same way as the minimization problem just by multiplying the objective function by -1 . By combining different properties of each arguments of the triplet, we can define a variety of optimization problems. A brief introduction to the traditional optimization method is given in Appendix B.

2.2 Metaheuristic Approaches to Optimization

In this section, we will review several emerging methods known as metaheuristic optimizations. Roughly speaking, metaheuristic optimizations are consid-

ered as a kind of direct search method aiming at a global optimum by utilizing a certain probabilistic drift and heuristic idea. The algorithms are commonly depicted as shown in Figure 2.1. To give a certain perturbation to the current (tentative) solution, a candidate solution will be generated. It is in turn evaluated through comparison with the tentative solution. Not only when the candidate is superior to the tentative (downhill move), but also when it is a bit inferior (uphill move), the candidate solution can become a new tentative solution with the prescribed probability. By occasionally accepting an inferior candidate (uphill move), these methods can escape from the local optimum and attain the global optimum as illustrated in Figure 2.2.

From these tactics, the algorithms are mainly characterized by the manners in which to derive the tentative, how to nominate the candidate, and how to decide the solution update. Metaheuristic optimization can also readily cope with even the combinatorial optimization. Due to these favorable properties and support by the outstanding progress both of computer hardware and software, these methods have been widely applied to solve difficult problems in recent manufacturing optimization [1, 2].

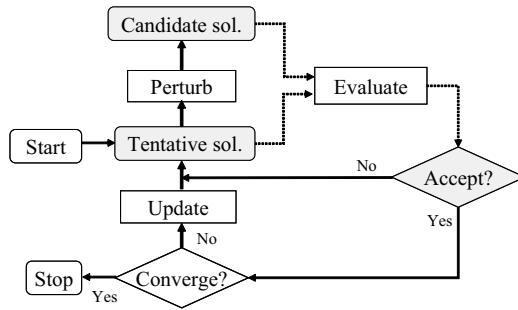


Fig. 2.1. General procedure of the metaheuristic approach

2.2.1 Genetic Algorithms

Genetic algorithm (GA) [3, 4, 13] is a pioneering method of metaheuristic optimization which originated from the studies of cellular automata of Holland [6] in the 1970s. It is also known as an evolutionary algorithm and a search technique that copies from biological evolution. In GA, a population of candidate solutions called individuals evolves toward better solutions from generation to generation. Since it needs no difficult mathematical conditions and can perform well with all types of optimization problems, it has been widely applied to solve problems not only in the engineering field but also in art, biology, economics, operations research, robotics, social sciences, and so on. The algorithm is closely related to some terminologies of natural selection,

on its fitness. Contrasting the iteration with the generation, the optimization process is defined by the search on a solution set $P_{\text{OP}}(t)$ described at the t -th generation as follows:

$$P_{\text{OP}}(t) = \{x_{1,t}, x_{2,t}, \dots, x_{N_p,t},\} \quad (2.1)$$

where N_p denotes a population size, and $x_{i,t}$, ($i = 1, 2, \dots, N_p$) is supposed to be a genotype. When we do need to note the generation explicitly, x_i means $x_{i,t}$ hereinafter.

At each generation, the fitness of whole population is evaluated, and the survival individuals are selected through a reproduction process where fitter solutions are more likely to be selected. Simply, the objective function is amenable to the fitness function of x_i , *i.e.*, $F_i = f(x_i)$, (≥ 0). To keep regularity and increase the efficiency, however, the original value should be transformed into the more proper value using a certain scaling technique. The following are typical scaling methods:

1. linear scaling
2. sigma truncation
3. power law scaling

In the above, linear scaling simply applies a linear transformation to F_i (≥ 0)

$$\hat{F}_i = aF_i + b,$$

where a and b are appropriately chosen coefficients. Sigma truncation is applied as

$$\hat{F}_i = aF_i - (\bar{F} - c\sigma),$$

where \bar{F} and σ denote the average of the fitness over the population and its standard deviation, respectively. Moreover, c is a parameter between $1 \sim 3$. Finally, power law scaling is described as

$$\hat{F}_i = (F_i)^k, \quad (k > 1).$$

Since the implementation and the evaluation of the fitness are important factors affecting the speed and efficiency of the algorithm, the scaling has a particular significance. Evolution or search takes place through genetic operators such as reproduction, mutation and crossover, each of which will be explained below.

A. Rule of Reproduction

As to why the rule of natural selection is applied to the optimization may rely on an observation that the better solutions often locate in the niche of good solutions found so far. This is compared to a concept regarding the stationary condition for optimization in a mathematical sense. The following rules are popularly known as the reproduction:

1. Roulette selection: This applies the rule that individuals can survive into the next generation based on the rate of fitness value of each (F_i) to the total value ($F_T = \sum_{k=1}^{N_p} F_k$), *i.e.*, $p_i = F_i/F_T$, as shown in Figure 2.3. This can constitute a rationale such that an individual with a greater fitness has a larger possibility of being selected in the next generation; an individual with even a low fitness has a chance of being selected. For these reasons, we can maintain the manifold of the population, and prevent it from being trapped at the local optimum. In addition, since this rule is simple, it is considered as a basic rule in the reproduction of GAs. There are two variants of this rule.

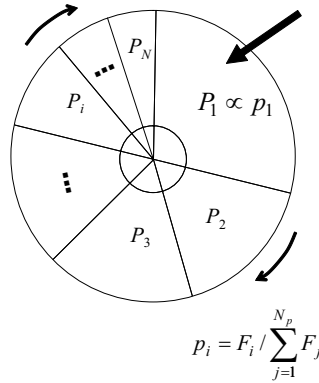


Fig. 2.3. Roulette selection

- Proportion selection: Generating a random value between $[0,1]$ (denoted by $\text{rand}()$), search the minimum k satisfying the condition such that $\sum_{i=1}^k F_i \geq \text{rand}() F_T$. Then the k -th individual can survive. This procedure is repeated until the total number of survivors becomes N_p .
 - Expected-value selection: The above methods sometimes cause an undue selection due to probabilistic drift when the number of population is not sufficiently large. To fix this problem, this method tries to select the individual in terms of the expected value based on the rate p_i . That is, when the required number of selections is N_s , the i -th individual can propagate by $[p_i N_s]$. Here $[\cdot]$ denotes the Gauss symbol.
2. Ranking selection: This method can fix a certain problem occurring with roulette selection. Let us consider a situation where there exist individuals with extremely high fitness values, or there is almost no difference among the fitness values of individuals. In the former case, it can happen that only the particular individuals will flourish, while in the latter every individual dwells on the average and the better ones cannot grow for ever. Instead of the magnitude of fitness itself, it is possible to achieve a proportional selection by paying attention to the ranking. According to the magnitude

of fitness, rank the individual first. Then the selection will take place in the order of the selection rate decided *a priori*. For example, linear ranking sets up the selection rate for the individual at the i -th place of the ranking as $p_i = a - b(i - 1)$ meanwhile non-linear one as $p_i = c(1 - c)^{i-1}$, where a , b , and c are coefficients in $(0, 1)$.

3. Tournament selection: In this method, the individuals with the highest fitness among the fixed size of the sub-population selected randomly will survive through tournament. This procedure is repeated until the pre-determined number of selections has been attained.
4. Elitist preserving selection: If we select by relying only on a probabilistic basis, favorable individuals happen to disappear due to the probability drift also imbedded in genetic operations like crossover and mutation. This phenomenon may cause a performance degradation known as premature convergence. Though this is the generic nature of GA, it has a side effect of preventing trapping at the local optimum. Noticing these facts, this selection preserves the elite in the present population without any reserve for the next generation. This has a certain effect of preventing that the best be killed through the genetic operations, but, in turn, produces the risk of another convergence. Consequently, this method should be applied together with another selection method. Obviously, under this rule the highest value of fitness increases monotonically along with the generation.

B. Crossover

This operation plays the most important role in GA. Through the crossover, a pair selected randomly from the population becomes parents, and produce a pair of offspring that share the characteristics of their parents by exchanging genes with each other. To use this mechanism, we need to define properly three routines: how to select the pairs, how to recombine the chromosome, and how to migrate the offspring into the population. Though various crossover methods have been proposed, depending on the problem, below we show only a few typical methods for the case of binary coding, *i.e.*, $\{0, 1\}$, for simplicity.

1. One-point crossover

Select randomly a crossover point in the string of parents and exchange the right-hand parts mutually. (Below “|” represents the crossover point)

Parent 1 :	01001 101	Offspring 1 :	01001 110
Parent 2 :	01100 110	Offspring 2 :	01100 101

2. Multi-point crossover

Select randomly plural crossover points in the string and exchange the parts mutually. (See below for the two-point crossover)

Parent 1 :	010 011 01	Offspring 1 :	010 001 01
Parent 2 :	011 001 10	Offspring 2 :	011 011 10

3. Uniform crossover

This method first prepares a mask pattern by generating $\{0, 1\}$ uniformly at every locus beforehand. Then offspring “1” inherits the character of parent “1” if the allele of the mask pattern is 1, and parent “2” if it is 0. Meanwhile, offspring “2” is generated in an opposite manner. See the following example, which assumes that the mask pattern is given as 01101101:

Parent 1 : 01001101	Offspring 1 : 01001111
Parent 2 : 01100110	Offspring 2 : 01100100

The simple crossover operates as follows:

- Step 1: Set $k = 1$.
- Step 2: Select randomly a pair of individuals (parents) from among the population.
- Step 3: Apply an appropriate crossover rule to the parent to produce a pair of offspring.
- Step 4: Replace the parent with the offspring. Let $k = k + 1$.
- Step 5: If $k > [p_C N_p]$, where p_C is a crossover rate, stop. Otherwise, go back to Step 2.

C. Mutation

Since the crossover produces offspring that only have characteristics from their parents, the manifold of the population is likely to be restricted within a narrow extent. A mutation operation can compensate this problem and keep the manifold by replacing the current allele with others with a given probability, say p_M . A simple flip-flop type mutation takes place such that: first select randomly an individual, select randomly a mutation point for the selected individual, reverse the bit thereat, and repeat until the number of this operation exceeds $[p_M N_p L]$. For example, when such a mutation point locates at the third place from the left-hand side, a change occurs for the gene of the selected individual.

Before : 01(1)01101	After : 01(0)01101
---------------------	--------------------

In addition to the above, varieties of mutation methods have been proposed so far. They are as follows:

1. Displacement: move part of the gene to another position of the same chromosome.
2. Duplication: copy some of the genes to another position.
3. Inversion: reverse the order of some genes in the chromosome.
4. Addition: insert some of the genes in the chromosome. This causes an increase in the length of the chromosome.
5. Deletion : delete some of the genes in the chromosome. This causes a decrease in the length of chromosome.

D. Summary of the Algorithm

The entire GA procedure is outlined in the following. The flow chart is shown in Figure 2.4.

- Step 1: Let $t = 0$. Generate N_p individuals randomly and define the initial population $P_{OP}(0)$.
- Step 2: Evaluate the fitness value for each individual. When $t = 0$, go to Step 3. Otherwise reproduce the individuals by applying an appropriate production rule.
- Step 3: Under the prescribed probabilities, apply crossover and mutation in turn. These genetic operations produce the updated population $P_{OP}(t+1)$.
- Step 4: Check the stopping condition. If it is satisfied, select the individual with highest fitness as a (near) optimal solution and stop. Otherwise, go back to Step 2 after letting $t := t + 1$.

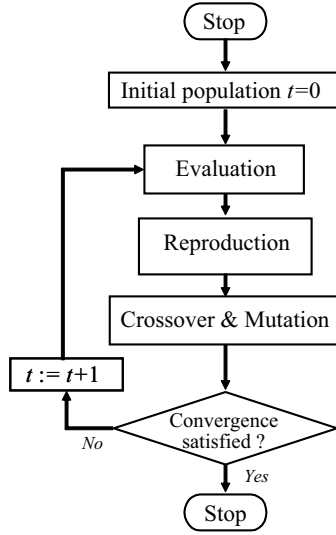


Fig. 2.4. Flow chart of the GA algorithm

Stopping conditions are commonly used as follows:

1. After a prescribed number of generations
2. When the highest fitness is not updated for a prescribed period
3. When the average fitness of the population has been almost saturated
4. A combination of the above conditions

Eventually, major factors of GA refer to the reproduction in Step 2 and the genetic operations in Step 3. In a word, the reproduction makes a point of

finding better solutions and concentrates on the search around these, while the crossover and mutation try to spread the search space via a stochastic perturbation and to avoid staying at the local optimum. With a better complement of these properties with each other, GA can be used as an efficient search technique. Since GA is problem specific, it is necessary to adjust parameters such as mutation rate, crossover rate and population size to find reasonable settings for the problem class being worked on. A very small mutation rate may lead to genetic drift or premature convergence in a local optimum. On the contrary, a mutation rate that is too high may lead to the loss of good solutions.

E. Miscellaneous

The building block hypothesis is a theoretical background that supports the effectiveness of GA [13, 6]. It says that short, low order, and highly fit schemata are sampled, recombined, and re-sampled to form strings of potentially higher fitness. In a way, by working with these particular schemata (the building blocks), we have reduced the complexity of our problem; instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings. This hypothesis requires coding to satisfy the following conditions.

- Individuals having similar phenotype are also close to each other regarding genotype.
- No major interference occurs between the loci.

From this aspect, Gray coding $(g_{l-1}, g_{l-2}, \dots, g_0)$ is known to be more favorable than binary coding $(b_{l-1}, b_{l-2}, \dots, b_0)$ because it can avoid the case where many simultaneous mutations or crossovers need to change the chromosome for a better solution. For example, let us assume that value 7 is optimal, and there exist the near optimal solutions with value 8. For these values, 4 bit binary coding of 7 is 0111 and 1000 for 8. Meanwhile, Gray coding becomes 0100 and 1100, respectively. Then, Gray coding can change 8 into 7 only by one mutation, but the binary coding needs such an operation four times successively. The following equation gives the relation between these types of coding:

$$g_k = \begin{cases} b_{l-1} & \text{if } k = l - 1 \\ b_{k+1} \oplus b_k & \text{if } k \leq l - 2 \end{cases} ,$$

where operator \oplus applies the exclusive disjunction.

By virtue of the nature related to multi-start algorithms, we can expect to attain the global optimum more easily and more certainly than with any conventional single-start algorithms. To make use of this advantage, keeping the manifold during the search is a special importance for GA. In a sense, this is closely related to the status of the initial population and the stopping condition. The following are a few other well-known tips:.

1. The initial population should be selected by extracting the best N_p among the individuals with more than the prescribed population size ($> N_p$).
2. Mutation may destroy the favorable schema that crossover has built (building block hypothesis). Hence parameters controlling these operations should be set as $p_C > p_M$, and additionally p_M is designed so as to decrease along with the generation.

When applying GA to the constrained optimization problem described as

$$[Problem] \quad \min f(x) \quad \text{subject to} \quad \begin{cases} g_i(x) \geq 0 & (i = 1, 2, \dots, m_1) \\ h_j(x) = 0 & (j = m_1 + 1, \dots, m), \end{cases}$$

the following penalty function approach is usually adopted:

$$f'(x) = f(x) + P \left\{ \sum_{i=1}^{m_1} \max[0, -g_i(x)] + \sum_{i=m_1+1}^m h_i(x)^2 \right\},$$

where $P(> 0)$ denotes the penalty coefficient.

The real number coding is better and provides higher precision for the problem with a large search space where the binary coding would require a prohibitively long representation. This coding is straightforward, and the real value of each variable corresponds directly to the gene of each chromosome. The crossover is defined arithmetically as a linear combination of two vectors. When P_1 and P_2 denote the parent solution vectors, the offspring are generated as $O_1 = aP_1 + (1-a)P_2$ and $O_2 = (1-a)P_1 + aP_2$, where a is a random value in $\{0, 1\}$. On the other hand, the mutation starts with randomly selecting an individual V . Then the mutation is applied in two ways, that is, simple mutation applies the following equation only for mutation point k appointed randomly in V , while the uniform mutation applies this to every locus:

$$V_k = v_k^L + r(v_k^U - v_k^L) \quad \text{where} \quad k = \begin{cases} \exists k : & \text{for simple mutation} \\ \forall k : & \text{for uniform mutation} \end{cases},$$

where v^U and v^L are the lower and upper bounds, respectively, and r is a random number from uniform probability distribution. A certain local search scheme is generally incorporated for these genetic operations to find a better solution near the current one.

2.2.2 Simulated Annealing

Simulated annealing (SA) is another metaheuristic algorithm specially suitable for the global optimization in terms of giving a certain probabilistic perturbation [7, 8]. It borrows the idea from a physical mechanism known as

annealing in metallurgy. Annealing is a popular engineering technique that applies heating and controlled cooling for material to increase the size of its crystals and reduce their structural defects. Heating causes the atoms to activate the kinetic energy, and is likely to make them unstuck at their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy. In contrast, slow cooling gives atoms more chances of finding configurations with lower internal energy than the initial one.

By analogy with this physical process, SA tries to solve the optimization problem. In its solution, each point of the search space is compared to a state of some physical system, and the objective function to be minimized is interpreted as the internal energy status of the system. When the system attains the state with the minimum energy, we can claim that the optimal solution has been obtained. Its basic iteration process is described as follows.

- Step 1: Generate an initial solution (let it be a current solution x), and also set an initial temperature T .
- Step 2: Consider some neighbors of the current state, and select randomly a neighbor x' in it as a possible solution.
- Step 3: Decide probabilistically whether to move on state x' or to stay at state x .
- Step 4: Check the stopping condition, and if it is satisfied, stop. Otherwise, cool the temperature and go back to Step 2.

The probability moving from the current solution to the neighbor in Step 3 depends on the difference between the respective objective function values and a time-varying parameter called temperature T . The algorithm is designed so that the current solution changes almost randomly when T is high, while the solution descends downhill as a whole with the decrease in temperature. The allowance for uphill moves during the process may avoid sticking at the local minima and make it possible to be a good approximation of the global optimum as illustrated in Figure 2.2. Let us describe the detail of the essential features of SA in the following.

A. Neighbors of State

Though the selection of neighbors (local search) has a great affect on the performance of the algorithm, no general methods have been proposed since they are very problem-specific. The concept of local search may be modeled conveniently as a search graph where vertices represent the states, and an edge denotes a transition between the vertices. Then, the length of a path represents the degree of the niche of neighbors, supposing the neighbors are expected to all have nearly the same energy. It is desirable to go from the initial state to a better state successively by a relatively short path on this graph, and such a path must be followed by the iteration of SA as similarly as possible.

Regarding the generation of neighbors, many ideas have been proposed for each class of problem so far, *i.e.*, the n -opt neighborhood and the or -opt neighborhood in the traveling salesman problem; the insertion neighborhood and the swap neighborhood in the scheduling problem; the λ -flip neighborhood in the maximum satisfiability problem, and so on [1].

B. Transition Probabilities

The transition from the current state x to a candidate state x' will be made according to the probability given by a function $p(e, e', T)$ where $e = E(x)$ and $e' = E(x')$ denote the energies of the two states (presently objective function values). An essential requirement for the transition probability is that $p(e, e', T)$ is non-zero when $e' \geq e$. This means that the system may move to the new state even if it is worse (has a higher energy) than the current one. The allowance for such uphill moves during the process may avoid sticking at the local minimum, and one can expect a good approximation of the global optimum as noted already.

On the other hand, as T tends to zero, the probability $p(e, e', T)$ also approaches zero when $e' \geq e$, while keeping a reasonable positive value when $e' < e$. As T becomes smaller and smaller; therefore, the system will increasingly favor downhill moves, and avoid the uphill moves. When T approaches 0, SA performs just like the greedy algorithm, which makes the move if and only if it goes downhill.

The probability function is usually chosen so that the probability of accepting a move decreases according to the increase in the difference of energies $\Delta e = e' - e$. Moreover, the evolution of x should be sensitive to Δe over a wide range when T is high and only within the small range when T is small. This means that small uphill moves are more likely to occur than large ones in the latter part of the search. To meet such requirements, the following Maxwell-Boltzmann distribution governing the distribution of energies of molecules in a gas is popularly used (see also Figure 2.5):

$$p = \begin{cases} 1 & \text{if } \Delta e \leq 0 \\ \exp(-\Delta e/T) & \text{if } \Delta e > 0 \end{cases} .$$

C. Annealing Schedule

Another essential feature of SA is how to reduce the temperature gradually as the search proceeds. This procedure is known as the annealing (cooling) schedule. Simply speaking, the initial temperature is set to a high value so that the uphill and downhill transition probabilities become nearly the same. To do this, it is necessary to estimate Δe for a random state and its neighbors over the entire search space. However, this needs some amount of preliminary experiments. A more common method is to decide the initial temperature so

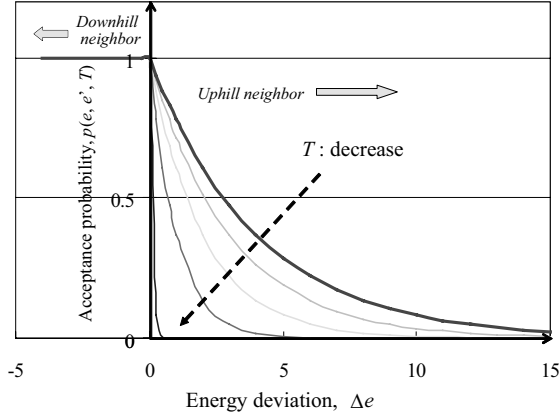


Fig. 2.5. The demanding character of probability function

that the acceptance rate in the search at the earlier stage will be greater than a prescribed value.

The temperature must decrease to zero or nearly zero by the end of the iteration. This is the only condition required for the cooling schedule, and many methods have been proposed so far. Among them, geometric cooling is a simple but popular method, in which the temperature is decreased by a fixed rate at each step, *i.e.*, $T := \beta T$, ($\beta < 1$). Another one termed exponential cooling is applied as $T = T_0 \exp(-at)$, where T_0 and t denote an initial temperature and iteration number, respectively. A more sophisticated method involves a heat-up step when the tentative solution has not been updated at all during a certain period. By returning the current temperature to the previous one, it tries to break the plateau status. In this way, the initial search makes a point of wandering a broad space that may contain good solutions while ignoring small degradations of the objective function. Then it will drift towards the low energy regions that become narrower and narrower, and finally aim at the minimum according to the descent strategy.

D. Convergence Features

It is known that the probability of finding the global optimal solution by SA approaches 1 as the annealing schedule is continued infinitely. This theoretical result is not helpful for deciding a stopping condition in practice. The simplest condition is to terminate the iterations after the prescribed number for which the temperature is reduced nearly by zero according to the annealing schedule. Various methods can be considered by observing the status of convergence more elaborately in terms of an update of the tentative solution.

Sometimes it is better to move back to a solution that was significant rather than always moving from the current state. This procedure is called

restarting. The decision to restart could be made based on a fixed number of steps, or on the current solution being too poor compared with the best one obtained so far.

Finally, applying SA to a specific problem, we must specify the state space, the neighbor selection method, the probability transition function, and the annealing schedule. These choices can have a significant impact on the effectiveness of the method. Unfortunately, however, there is neither specific value that will work well with all problems, nor a general way to find the best setting for a given problem.

2.2.3 Tabu Search

Though tabu search (TS) [9, 10] has a simple solution procedure, it is an effective method for combinatorial optimization problems. In a word, TS belongs to a class of local search techniques that enhances performance by using a special memory structure known as the tabu list. TS repeats the local search iteratively to move from a current solution x to a possible and best solution x' in the neighbor of x , $N(x)$. Unfortunately, there exists the case where simple local search may cause a cycling of the solution, *i.e.*, from x to x' , and from x' to x . To avoid such cycling, TS use the tabu list that corresponds to a short term memory cited in the field of recognition science. Transition to any solutions involved in the tabu list is prohibited for a while, even if this will provide an improvement of the current solution. Under such restrictions, TS continues the local search until a certain stopping condition has been satisfied. The basic iteration process is outlined as follows:

- Step 1: Generate an initial solution x and let $x^* := x$, where x^* denotes the current best solution. Set $k = 0$ and let the tabu list $T(k)$ be empty.
- Step 2: If $N(x) - T(k)$ is empty, stop. Otherwise, set $k := k + 1$ and select x' such that $x' = \min f(x)$ for $\forall x \in N(x) - T(k)$.
- Step 3: If x' outperforms the current solution x^* , *i.e.*, $f(x') \leq f(x^*)$, let $x^* := x'$.
- Step 4: If a chosen number of iterations has elapsed either in total or since x^* was last improved, stop. Otherwise, update $T(k)$, and go back to Step 2.

In the TS algorithm, the tabu list plays the most important role. It makes it possible to explore the search space that would be left unexplored and to escape from the local optimum. The simplest form of the tabu list is a list of the solutions by the latest m -visits. Referring to this list, transition to the solutions recorded in the tabu list is prohibited to move during a period of m length. Such period is called the tabu tenure. In other words, the validity of such prohibition holds only during the tabu tenure, and its length can control the regulation regarding the transition. That is, if it is long, then the transition is hardly restricted and *vice versa*.

Other structures of the tabu list utilize certain attributes associated with the particular search technique depending on the problem. Solutions with such attributes are labeled to be tabu-active, and the tabu-active solutions are also viewed as tabu for the search. For example, in the traveling salesman problem (TSP), solutions that include certain arcs are prohibited or an arc that was added newly to a TSP tour cannot be removed in the next m -moves. Generally speaking, tabu lists containing the attributes are much more effective. However, by forbidding the solutions that contain tabu-active elements, more than one solution is likely to be declared as the tabu. Hence, there exist the cases where some solutions might be avoided although they have excellent quality and have not yet been visited.

Aspiration criteria serve to relax such restrictions. They allow overriding the tabu state of the solutions that are better than the currently best known solution, and keep it in the allowed set. Besides these special ideas, a variety of extensions are known, some of which are cited below.

The load of local search can be reduced if we concentrate the search only on the promising extent instead of whole neighbor. Such an idea is generally called a candidate list strategy. After selecting k -best solutions among the neighbor solutions, probabilistic tabu search is to replace the current solution randomly with one depending on the probabilities, which are decided based on their objective functions. This idea is very similar to the roulette strategy in the selection of GA.

In addition to the function of the tabu list as a short-term memory, a long-term memory is available to improve the performance of the algorithm. This generic name refers to an idea that tries to utilize the history of information along with the search process. The long-term memory makes it possible to use the intensification of promising search and diversification for global search at the same time. For example, a transition measure in frequency-based memory records the numbers of the modification of the variables, while a residence measure records the number staying at the specific value. Since the high transition measure foresees the long-term search cycle, an appropriate penalty should imposed on its selection. On the other hand, the residence measure is available for the selection of initial solutions by controlling the appearance rate of the certain variables. That is, restriction of the variables with high measure can facilitate the diversification while promoting the intensification.

2.2.4 Differential Evolution (DE)

Differential Evolution (DE) is viewed as a real number coding version of GA and was developed by Price and Storn [11]. Though it is a very simple population-based optimization method, it is known as a very powerful method for real world applications. A variety of variants are classified using a triplet expression like $DE/x/y/z/$, where

- x specifies the method for selecting the parent vector to become a base of the mutant vector. Two selections, *i.e.*, chosen randomly (“rand”) or

chosen from the best in the current population (“best”) are typically employed.

- y is a number of the difference vector used in Equation 2.2.
- z denotes a crossover method. In binominal crossover (“bin”), crossover is performed on each gene of a chromosome while, in exponential crossover (“exp”) it is performed on a chromosome as a whole:

	$i = 1$	$i = 2$	\dots	$i = N_p$
$j = 1$	231	1121		781
	517	450		208
	\vdots	\vdots	\dots	\vdots
\vdots	976	0		432
	950	838		1000
	3200	3124		2945
$j = n$	873	1288		690

where N_p = population size
 n = number of decision variables

Fig. 2.6. Example of coding in DE

As is usual with every variant, users need the following settings before optimizing their own problem: the number of population N_p , scaling factor F and crossover rate p_C . The algorithm in the case of DE/rand/1/bin/ is outlined as follows:

Step 1 (Generation): Generate randomly every n -dimensional “target” vector to yield the initial population.

$$P_{OP}(t) = \{x_{i,t}\} \quad (i = 1, 2, \dots, N_p),$$

where t is a generation number and N_p is a population size. An example of coding is shown in Figure 2.6.

Step 2 (Mutation): Create each “mutant” vector by adding the weighted difference between two target vectors to the third target vector. These three vectors are chosen randomly among the population,

$$v_{i,t+1} = x_{r3,t} + F(x_{r2,t} - x_{r1,t}) \quad (i = 1, 2, \dots, N_p), \quad (2.2)$$

where F is real and constant in $[0, 2]$.

Step 3 (Crossover): Apply the crossover operation to generate the trial vector u_i by mixing some elements of the target vector with the mutant vector through comparison between the random value and the crossover rate (see also Figure 2.7),

$$u_{ji,t+1} = \begin{cases} v_{ji,t+1} & \text{if } \text{rand}(j) \leq p_C \text{ or } j = \text{rand}() \\ x_{ji,t} & \text{if } \text{rand}(j) > p_C \text{ and } j \neq \text{rand}() \end{cases} \quad (j = 1, 2, \dots, n),$$

where $\text{rand}(j)$ is the j -th evaluation of a uniform random number generator, p_C is the crossover rate in $[0, 1]$, and $\text{rand}()$ is a randomly chosen index in $\{1, 2, \dots, n\}$. Ensure that $u_{i,t+1}$ has at least one elements from the mutant vector $v_{i,t+1}$. Then evaluate the performance of each vector.

Step 4 (Selection): If the trial vector outperforms the target vector, the target vector is replaced with the trial vector. Otherwise, the target vector is retained. Thus, the members of the new population for the next generation are selected in this step.

Step 5: Check the stopping condition. If it is satisfied, stop and return the overall best vector as the final solution. Otherwise, go back to Step 2 by incrementing the generation number by 1.

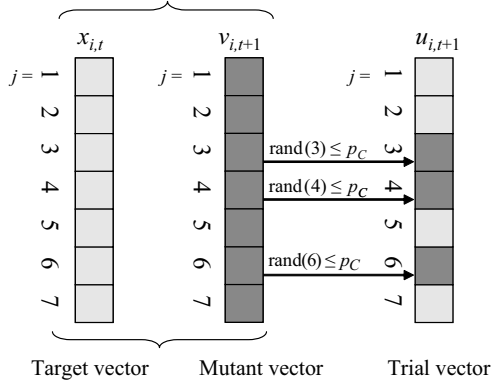


Fig. 2.7. Crossover operation of DE

In the case of DE/best/2/bin, at the above Step 2, the mutant vector is derived from the following equation:

$$v_{i,t+1} = x_{\text{best},t} + F(x_{r1,t} + x_{r2,t} - x_{r3,t} - x_{r4,t}) \quad (i = 1, 2, \dots, N_p),$$

where $x_{\text{best},t}$ is the best solution at generation t . Moreover, the exponential crossover in Step 3 is applied as

$$u_{ji,t+1} = \begin{cases} v_{ji,t+1} & \text{if } \text{rand}() \leq p_C \\ x_{ji,t} & \text{if } \text{rand}() > p_C \end{cases} \quad (\text{for } \forall j).$$

For successful application of DE, there are several tips regarding parameter setting and tuning, some of which will be shown below.

1. The number of population N_p is normally set between five to ten times the number of decision variables.
2. If a proper convergence cannot be attained, it is better to increase N_p , or adjust F and p_C both in the range $[0.5, 1]$ for most problems.
3. Simultaneous increase in N_p and decrease in F make the convergence more likely to occur but generally make it longer.
4. DE is much more sensitive to the choice of F than p_C . Though larger p_C gives faster convergence, it is sometimes necessary to use a smaller value to make DE robust enough for the particular problem. Thus, there is always a tradeoff between convergence speed and robustness.
5. p_C of binominal crossover should usually be set higher than that of the exponential crossover.

A. Adaptive DE

To improve the convergence, a variant of DE (ADE) was proposed recently¹. It introduced ideas of a gradient field in the objective function space and an age for individuals to control the crossover factor. The algorithm is outlined below.

- Step 1(Generation): Reset the generation at 1 and the age at 0. $Age(i)$ is defined as the number of generations during which each individual i is alive. Then generate $2N_p$ individuals x_i in n -dimensional space.
- Step 2 (Gradient field): Make a pair randomly for each individual and compare their objective function values. Then, classify them into winner (having smaller value) and loser, and register as winner and loser, respectively. The winners will age by one, and the losers rejuvenate by one.
- Step 3 (Mutation): Pick up randomly a base vector $x_{\text{base}()}$ from the winner. Moreover, choose randomly a pair building the gradient field and generate a mutant vector as follows:

$$v_{i,t+1} = x_{\text{base},t} + F(x_{\text{better}(),t} - x_{\text{worse}(),t}) \quad (i = 1, \dots, 2N_p),$$

where $x_{\text{better}()}$ and $x_{\text{worse}()}$ denote the winner and loser of each pair, respectively. This operation may generate mutants in the direction possible for decreasing the objective function globally everywhere in the search space.

¹ Shimizu Y (2005) About adaptive DE. *Private Communication*

Step 4 (Crossover): The same type of crossover as has already been mentioned is available. However, its rate p_C will be decided by a monotonic decreasing function of age, *e.g.*,

$$p_C = (a + c)e^{-b \cdot \text{Age}(i)} + c, \quad \text{or} \quad p_C = \max[a + c - b \cdot \text{Age}(i), c],$$

where a, b and c are real positive constants to be determined by the user under the condition that $0 < a + c < 1$ (see 2.8). This crossover rate makes the target vectors that have lived for long time (having an older age) more likely to survive in the next generation.

Step 5 (Selection): If the trial vector is better than the target vector, replace the target vector with the trial vector and give it a new age suitably *e.g.*, reset (0). Otherwise, the target vector is retained and it gets older by one.

Step 6: Check the stopping condition. If it is satisfied, stop and return to the overall best vector as the final solution. Otherwise, go back to Step 2 by updating the generation.

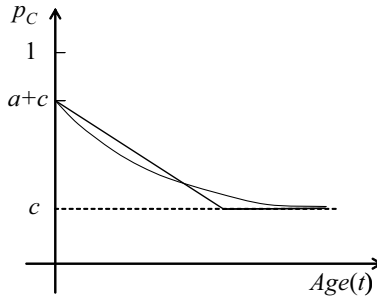


Fig. 2.8. Crossover rate depending on age

The following simple test problem validates the effectiveness of this method. Minimization of the Rosenbrock function is compared with the conventional method DE/rand/1/bin/:

$$f(x) = 100 \cdot (x_1^2 - x_2)^2 + (1 - x_1)^2, \quad x_1, x_2 \in [-10, 10].$$

Although, there are only two decision variables, this problem has the reputation of being a difficult minimization problem. The global minimum is located at $(x_1, x_2) = (1, 1)$. The comparison of convergence features between ordinal and adaptive DE is shown in Figure 2.9 in the logarithm scales. The linear model of age is used to calculate p_C as $p_C = 0.5 \cdot \max[1 - 0.0001 \cdot \text{Age}(i), 0.5]$. The adaptive method (“DE-rev”) is known to present a good convergence feature compared with the conventional method (“DE-org”).

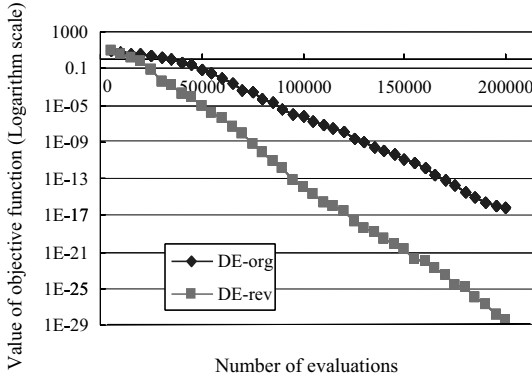


Fig. 2.9. Comparison of convergence features

2.2.5 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) , which was developed by J. Kennedy [12], is also a real number coding metaheuristic method for optimization. It is a form of swarm intelligence in the artificial intelligence study of the collective behavior in decentralized and self-organized systems. It stems from the theory of boids by C. Reynolds [13]. Imagining the behavior of a swarm of insects or a school of fish, we can observe that when one member finds a desirable path to go, (*i.e.*, for food, protection, *etc.*), the rest of the swarm can follow it quickly even if they are on the opposite side of the swarm.

The algorithm of PSO relies on the strength that such behavior to attain the goal is rational, and can be simulated by only three movements termed separation, alignment, and cohesion.

- Separation is a rule to separate one object from a neighbor, and prevent from colliding with each other. For this purpose, a boid flying ahead must speed up while those in the rear slow down. Moreover, the boids can change direction to avoid obstacles.
- By alignment, all objects try to adapt their movement to the others. Front boids flying far away will slow down and the rear boids will speed up to catch up.
- Cohesion is a centripetal rule for not disturbing the shape of the population as a whole. This requires boids to fly to the center of the swarm or the gravity point.

According to these three movements, PSO can be developed by imaging boids with a position and a velocity. These boids fly through hyperspace and remember the best position that they have seen. Members of a swarm communicate with each other and adjust their own position and velocity based on the information regarding the good positions both of their own (local bests)

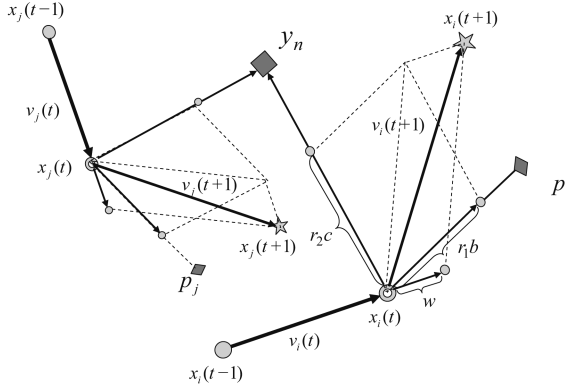


Fig. 2.10. Search scheme of PSO

and a swarm best (global best) as depicted in Figure 2.10. Updating of the position and the velocity is done through the following formulas:

$$x_i(t+1) = x_i(t) + v_i(t+1), \quad (2.3)$$

$$v_i(t+1) = w \cdot v_i(t) + r_1 b(p_i - x_i(t)) + r_2 c(y_n - x_i(t)) \quad (i = 1, 2, \dots, N_p), \quad (2.4)$$

where

t is the generation,

N_p is the population size (number of boids),

w is an inertial constant (usually slightly less than 1),

b and c are constants making a point of how much the boid is directed toward the good position (usually around 1),

r_1 and r_2 are random values in the range $[0,1]$,

p_i is the best position seen by the boid i ,

y_n is the global best position seen by the swarm.

The algorithm is outlined below.

Step 1: Set $t = 1$.

Initialize $x(t)$ and $v(t)$ randomly within the range of these values.

Initialize each p_i to the current position.

Initialize y_n to the position that has the best fitness among swarms.

Step 2: For each boid, do the following:

obtain $v_i(t+1)$ according to the Equation 2.4,

obtain $x_i(t+1)$ according to the Equation 2.3,

evaluate the new position,
 if it outperforms p_i , update it,
 if it outperforms y_n , update it.

Step 3: If the stopping condition is satisfied, stop. Otherwise let $t := t + 1$, and go back to Step 2.

2.2.6 Other Methods

In what follows, a few useful methods will be introduced. Generally speaking, they can exhibit advantages over the methods mentioned above for a particular class of problems. Moreover, they are amenable for various hybrid approaches of metaheuristic methods relying on the features characterized by probabilistic deviation, multi-modality, population-base, multi-start, *etc.*

The ant colony algorithm (ACO) [14, 15] is a probabilistic optimization technique that mimics the behavior of ants finding paths from the colony to food. In nature, ants wander randomly to find food. On the way back to their colony, they lay down pheromone trails. If other ants find such trails, they can reach the food source more easily by following the trail. Hence, if one ant can find a good or short path from the colony to the food source, other ants are more likely to follow that path. Since the pheromone trail evaporates with time, its attractive strength will gradually reduce. The more time it takes for an ant to travel, the more pheromones will evaporate. Since a short path is traced faster, the pheromone density remains high. Such positive feedback eventually makes all the ants follow a single path. Pheromone evaporation has also the advantage of avoiding the convergence to a local optimum. ACO has an advantage over SA and GA when the food source may change dynamically, since it can adapt to the changes continuously. Moreover, this idea is readily available for applying a multi-start technique in various metaheuristic optimizations.

Memetic algorithm [16] is an approach emerging from traditional GA. By combining local search with the crossover operator, it can provide considerably faster convergence, say orders of magnitude, than traditional GA. For this reason, it is called genetic local search or the hybrid genetic algorithm. Moreover, it should be noticed that this algorithm is most suitable for parallel computing.

An evolutionary approach called scatter search [17] is very different from the other evolutionary methods. It possesses a strategic design mechanism to generate new solutions while other approaches resort to randomization. For example, in GA, two solutions are randomly chosen from the population and crossover or a combination mechanism is applied to generate one or more offspring. Scatter search works based on a set of solutions called the reference set, and combines these solutions to create new ones based on the generalized path constructions in Euclidean space. That is, by both convex (linear) and

non-convex combination of two different solutions, the reference set can evolve in turn (reference set update)².

In Figure 2.11 it is assumed that the original reference solution set consists of the circles labeled A, B and C (diversified generation, enhancement). In terms of a convex combination of reference solutions A and B (solution combination), a number of solutions in the line segment defined by A and B may be created (subset generation). Among them, only solution 1 that satisfies a certain criteria for membership is involved in the reference set. In the same way, convex and non-convex combinations of original and new reference solutions create points 2, 3 and 4, one after another. After all, the resulting reference set consists of seven solutions in the present case. Unlike a “population” in GA, the number of reference solutions is relatively small in scatter search. Scatter search chooses only two or more reference solutions in a systematic way to create new solutions as shown above.

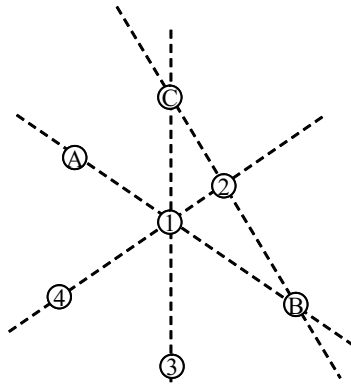


Fig. 2.11. Successive generation of solutions by scatter search

The following five major features characterize the implementation of scatter search.

1. Diversified generation: to generate a set of diverse trial solutions using an arbitrary initial solution (or seed solution).
2. Enhancement: to transform a trial solution into one or more improved trial solutions.
3. Reference set update: to build and maintain a reference set consisting of the k -best solutions found (where the value of k is typically small, *e.g.*, no more than 20). Solutions gain membership to the reference set according to their quality or their diversity.
4. Subset generation: to produce a subset of its solutions as a basis for creating combined solutions.

² This is similar to the movement of the simplex method stated in Appendix B.

5. Solution combination: to transform a given subset of solutions into one or more combined solution vectors.

In the sense that this method will rely on the reference solutions, this idea can also be used for applying the multi-start technique in some metaheuristic approaches.

2.3 Hybrid Approaches to Optimization

Since the term “hybrid” has broad and manifold meanings, we can give several hybrid approaches even if discussion might be restricted within the optimization methods. In what follows, three types of hybrid approach will be presented in terms of the combination of traditional mathematical programming (MP) and recent metaheuristic optimization (meta).

The first category is a “MP–MP” class. Most gradient methods for multi-dimensional optimization involve the optimization of step size search along the selected direction in the course of iteration. For this search, a scalar optimization method like the golden section algorithm or the Fibonacci algorithm is commonly used. This is a plain example of the hybrid approach in this class. Using an LP-relaxed solution as an initial solution and applying nonlinear programs (NLP) at the next stage may be another example of this class.

The second class “meta–meta” mainly appears in the extended or sophisticated application of the original algorithm of the metaheuristic method. Using the ACO method as the restarting technique of another metaheuristic method is an example of this class. Combining a binary code GA with other real number coding meta-methods is a reasonable way to cope with mixed-integer programs (MIP) . Instead of applying each method individually to solve MIP, such a hybrid approach can bring about a synergic effect to reduce the search space (chromosome length) and to improve the accuracy of the resulting solution (size of grains or quantification).

After all, many practical hybrid approaches may belong to the third “meta–MP” class. As supposed from the memetic algorithm or genetic local search, the local search is considered to be a promising technique that can accelerate the efficiency of the search compared with the single use of the metaheuristic method. Every method using an appropriate optimization technique for such local search may be viewed as a hybrid method in this class.

A particular advantage of this class will be exhibited to solve the following MIP in a hierarchical manner:

$$[Problem] \quad \min_{x,z} f(x, z)$$

$$\text{subject to } \begin{cases} g_i(x, z) \geq 0 & (i = 1, 2, \dots, m_1) \\ h_i(x, z) = 0 & (i = m_1 + 1, \dots, m) \\ x \geq 0, & (\text{real}) \\ z \geq 0, & (\text{integer}) \end{cases}.$$

This approach can achieve a good match not only between the upper and lower level problems but also each problem and the respective solution method. The most serious difficulties in solving MIP problems refer to the combinatorial nature in solution. By pegging the integer variables at the values decided at the upper level, the resulting lower level problem is reduced to a usual (non-combinatorial) problem that it is possible to be solved reasonably by MP. On the other hand, the upper level problem becomes an unconstrained integer programs (IP), and it is treated effectively by the metaheuristic method. Based on such an idea, the following hierarchical formulation is known to be amenable to solving MIP in a hybrid manner of “meta-MP” type (see also to Figure 2.12):

$$\begin{aligned} [Problem] \quad & \min_{z \geq 0: \text{integer}} f(x, z) \\ & \text{subject to } \min_{x \geq 0: \text{real}} f(x, z), \\ & \text{subject to } \begin{cases} g_i(x, z) \geq 0 & (i = 1, \dots, m_1) \\ h_i(x, z) = 0 & (i = m_1 + 1, \dots, m) \end{cases}. \end{aligned}$$

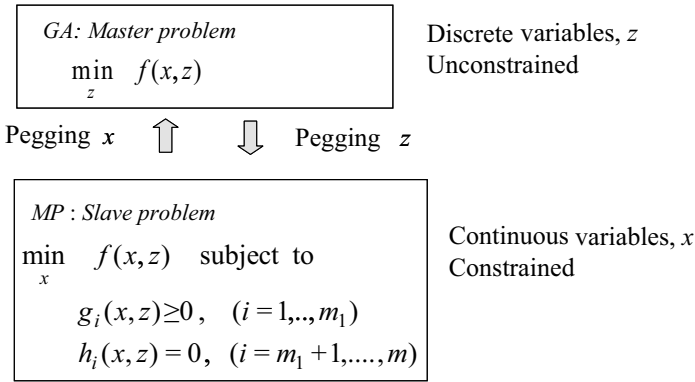


Fig. 2.12. Configuration of hybrid GA

In the above, the lower level problem becomes the usual mathematical programming problem. When the constraints of pure integer variables are involved, a penalty function method is available at the upper level as follows:

$$\min_{z \geq 0: \text{integer}} f(x, z) + P \left\{ \sum_i \max[0, -g_i(z)] + \sum_i h_i(z)^2 \right\}.$$

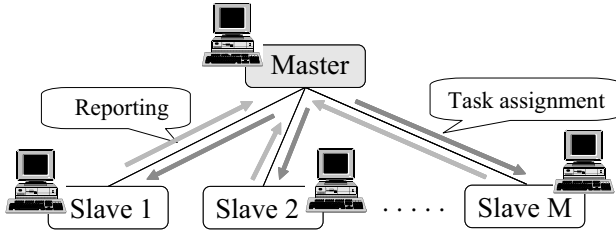


Fig. 2.13. Master-slave configuration for parallel computing

Moreover, by noticing the analogy of the above formulation to the parallel computing of the master-slave configuration as shown in Figure 2.13, an effective parallel computing is readily implemented [32]. There are many combinatorial optimization problems formulated as IP and MIP at every stage of the manufacturing optimization. The scheme presented here has close connections to various manufacturing optimization problems for which we can deploy this approach in an effective manner. For example, a large-scale network design and a site location problem under multi-objective optimization will be developed in the following sections.

2.4 Applications for Manufacturing Planning and Operation

Recent innovations in information technology as well as advanced transportation technologies are accelerating globalization of markets outstandingly. This raises the importance of just-in-time and agile manufacturing much more than before, since its effectiveness is pivotal to the efficiency of the business process. From this point of view, we will present three applications ranging from strategic planning to operational scheduling. We will also show how effectively the optimization problem in each topic can be solved by the relevant method employed there.

The first topic takes a logistic problem associated with supply chain management (SCM) [19, 20, 21]. It will be formulated as a hub facility location and route selection problem attempting to minimize the total management cost over the area of interest. This kind of problem [22, 23, 24] is also closely related to the network design of hub systems popular in various fields such as transportation [25], telecommunication [26], *etc.* However, most previous studies have scarcely called attention to the entire system composed both of distribution and collection networks. To deal with such large-scale and complex problems practically, an approach that decomposes the problem into sub-problems and applies a hybrid tabu search method will be described [27].

In terms of the small-lot-multi-kinds production, the introduction of mixed-model assembly lines is becoming popular in manufacturing. To increase the efficiency of such line handling, it is essential to prevent various

line stoppages incurred due to unexpected inconsistencies [28, 29]. The second topic concerns an injection sequencing problem for the manufacturing represented by the car industry [30]. The mixed-model assembly line thereat includes a painting line where we need to pay attention to uncertainties associated with so-called defective products. After formulating the problem, SA is employed to solve the resulting combinatorial optimization problem in a numerically effective manner.

The scheduling problem is one of the most important problems associated with the effective operation of manufacturing systems. Consequently, much of research has been done [31, 32, 33, 34], but most work only describes simple models [35]. Additionally, it should be noticed that the roles of human operators are still important although automation is now becoming popular in manufacturing. However, little research has taken into account the role of operators and the cooperation between operators and resources [36]. The third topic concerns a production scheduling managed by multi-skilled human operators who can manipulate multiple types of resources such as machine tools, robots, and so on [37]. After formulating a general scheduling problem associated with human tasks, a practical method based on a dispatching rule or an empirical optimization will be presented.

2.4.1 Logistic Optimization Using Hybrid Tabu Search

Recently, industries have been paying keen attention to SCM and studying it from various aspects [38, 39, 40]. It is viewed as a reengineering method managing life cycle activities of a business process to deliver added-value products and service to customers. As an essential part of decision making in such business processes, we consider a logistic optimization associated with a supply chain network(SCN) [27]. It is composed of suppliers, collection centers (CCs), plants, distribution centers (DCs), and customers as shown in Figure 2.14. Though CC can receive materials from multiple suppliers due to risk aversion (multiple allocation), each customer will receive products only from one DC (single allocation) that can deliver products either from another DC or customer. The problem is formulated under the conditions that the capacity of the facility is constrained, and demand, supply and per unit transport cost are given *a priori*. It refers to a nonlinear mixed-integer programming problem (MINLP) simultaneously deciding the location of hub centers and routes to meet the demands of all SCN members while minimizing the total cost,

$$\begin{aligned} \min \quad & \sum_{i \in I} \sum_{j \in J} D_i C1_{ij} r_{ij} + \sum_{j \in J} \sum_{j' \in J} \left(\sum_{i \in I} D_i r_{ij} \right) C2_{jj'} s_{jj'} \\ & + \sum_{j' \in J} \sum_{k \in K} \left(\sum_{j \in J} \left(\sum_{i \in I} D_i r_{ij} \right) s_{jj'} \right) C3_{j'k} t_{j'k} \end{aligned}$$

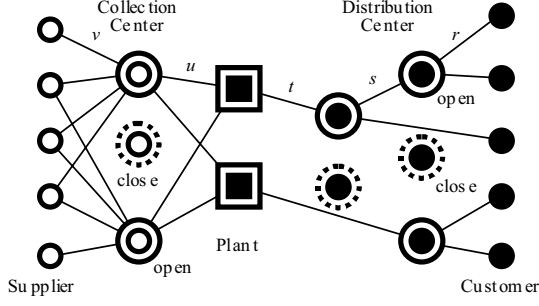


Fig. 2.14. Supply chain network

$$+ \sum_{k \in K} \sum_{l \in L} C4_{kl} u_{kl} + \sum_{l \in L} \sum_{m \in M} C5_{lm} v_{lm} + \sum_{j \in J} F1_j x_j + \sum_{l \in L} F2_l y_l,$$

subject to

$$\sum_{j \in J} r_{ij} = 1, \quad \forall i \in I, \quad (2.5)$$

$$\sum_{i \in I} D_i r_{ij} \leq P_j x_j, \quad \forall j \in J, \quad (2.6)$$

$$\sum_{j' \in J} s_{jj'} = x_j, \quad \forall j \in J, \quad (2.7)$$

$$\sum_{j \in J} \left(\sum_{i \in I} D_i r_{ij} \right) s_{jj'} \leq P_{j'} s_{j'j} x_{j'}, \quad \forall j' \in J, \quad (2.8)$$

$$\sum_{k \in K} t_{j'k} = s_{j'j'}, \quad \forall j' \in J, \quad (2.9)$$

$$\sum_{j' \in J} \left(\sum_{j \in J} \left(\sum_{i \in I} D_i r_{ij} \right) s_{jj'} \right) t_{j'k} \leq Q_k, \quad \forall k \in K, \quad (2.10)$$

$$\sum_{l \in L} u_{kl} = \sum_{j' \in J} \left(\sum_{j \in J} \left(\sum_{i \in I} D_i r_{ij} \right) s_{jj'} \right) t_{j'k}, \quad \forall k \in K, \quad (2.11)$$

$$\sum_{k \in K} u_{kl} \leq s_l y_l, \quad \forall l \in L, \quad (2.12)$$

$$\sum_{k \in K} u_{kl} = \sum_{m \in M} v_{lm}, \quad \forall l \in L, \quad (2.13)$$

$$\sum_{l \in L} v_{lm} \leq T_m, \quad \forall m \in M, \quad (2.14)$$

$r, s, t \in \{0, 1\}, \quad x, y \in \{0, 1\}, \quad u, v \in \text{real number},$

where binary variables x_i and y_i take 1 if each center i is open, and r_{ij} , s_{ij} , t_{ij} become 1 if there exist routes between customer i and DC j , DC i and DC j , and DC i and plant j , respectively. Otherwise, they are equal to 0 in all cases. u_{ij} and v_{ij} denote the amount of shipping from CC j to plant i and from supplier j to CC i , respectively. Moreover, D_i is the demand of customer i , and P_i , Q_i , S_i and T_i represent capacities of DC, plant, CC and supplier, respectively.

On the other hand, the first to fifth terms of the objective function are related to transport costs while the sixth and seventh terms to fixed charge costs of DC and CC, respectively. Equations 2.5, 2.7, and 2.9 mean that each customer, DC and plant are allowed to select only one location each in the downstream network. Equations 2.6, 2.8, 2.10, 2.12, and 2.14 represent the capacity constraints on the first stage DCs and the second stage DCs, plant, CC, and supplier, respectively. Equations 2.11 and 2.13 represent balance equations between input and output of plant and CC, respectively.

A. Hierarchical Procedure for Solution

(1) Decomposition into Sub-models

Since the solution of MINLP belongs to an NP-hard class, developing a practical solution method is more desirable than aiming at a rigid optimum. Noting the particular structure of the problem as illustrated in Figure 2.15, we can decompose the original SCN into two sub-networks originating from the plants in opposite direction to each other, *i.e.*, upstream (procurement) chain, and downstream (distribution) chain. The former solves a problem of how to supply raw materials from suppliers to plants via CCs, while the latter concerns how to distribute the products from plants to customers via DCs.

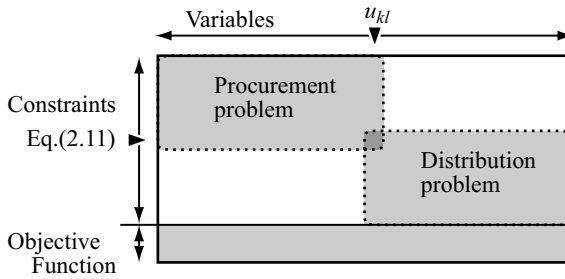


Fig. 2.15. A pseudo-block diagonal pattern of the problem structure

Eventually, to obtain a consequent result for the entire supply chain from what is solved individually, it is necessary to combine them consistently by adjusting a coupling constraint effectively. Instead of using Equation 2.11

directly as the coupling constraint, it is transformed into a suitable condition so that the tradeoff between the sub-networks can be adjusted through an auction-like mechanism based on an imaginary cost. For this purpose, we define the optimal cost associated with the procurement in the upstream chain C_{proc}^* ,

$$\sum_{k \in K} \sum_{l \in L} C4_{kl} u_{kl} + \sum_{l \in L} \sum_{m \in M} C5_{lm} v_{lm} + \sum_{l \in L} F2_l y_l = C_{\text{proc}}^*. \quad (2.15)$$

Then, dividing C_{proc}^* into each plant according to the amount of production, *i.e.*, $C_{\text{proc}}^* = \sum_k V_k$, we view V_k as an estimated shipping cost from each plant. Then, by denoting the unit procurement cost by ρ_k , we obtain the following equation:

$$\rho_k \sum_{j' \in J} \left(\sum_{j \in J} \left(\sum_{i \in I} D_i r_{ij} \right) s_{jj'} \right) t_{j'k} = V_k, \quad \forall k \in K. \quad (2.16)$$

Using this as a coupling condition instead of Equation 2.11, we can decompose the entire model into each sub-model as follows.

Downstream network (DC) model³

$$\begin{aligned} \min \quad & \sum_{i \in I} \sum_{j \in J} D_i C1_{ij} r_{ij} + \sum_{j \in J} \sum_{j' \in J} \left(\sum_{i \in I} D_i r_{ij} \right) C2_{jj'} s_{jj'} \\ & + \sum_{j' \in J} \sum_{k \in K} \left(\sum_{j \in J} \left(\sum_{i \in I} D_i r_{ij} \right) s_{jj'} \right) C3_{j'k} t_{j'k} + \sum_{j \in J} F1_j x_j, \end{aligned}$$

subject to Equations 2.5 - 2.10 and Equation 2.16.

Upstream network (CC) model

$$\min \quad \sum_{k \in K} \sum_{l \in L} C4_{kl} u_{kl} + \sum_{l \in L} \sum_{m \in M} C5_{lm} v_{lm} + \sum_{l \in L} F2_l y_l,$$

subject to Equations 2.12- 2.14 and Equation 2.17,

$$R_k = \sum_{l \in L} u_{kl} = \sum_{j' \in J} \left(\sum_{j \in J} \left(\sum_{i \in I} D_i r_{ij}^* \right) S_{jj'}^* \right) t_{j'k}^*, \quad (2.17)$$

where an asterisk means the optimal value for the downstream problem.

(2) Coordination Between Sub-models

³ A few variant models are solved by taking a volume discount of transport cost and multi-commodity delivery into account [41].

If the optimal values of the coupling quantities, *i.e.*, V_k or R_k , were known *a priori*, we could derive a consistent solution straightforwardly by solving each sub-problem individually. However, since this is not obviously expected, we need to make an adjusting process as follows.

- Step 1: For tentative V_k (initially not set forth), solve the downstream problem.
 Step 2: After calculating R_k based on the above result, solve the upstream problem.
 Step 3: Reevaluate V_k based on the above upstream optimization.
 Step 4: Repeat until no more change in V_k has been observed.

In addition, we rewrite the objective function of the downstream problem by relaxing the coupling constraint in terms of the Lagrange multiplier as follows:

$$\begin{aligned} & \sum_{i \in I} \sum_{j \in J} D_i C1_{ij} r_{ij} + \sum_{j \in J} \sum_{j' \in J} \left(\sum_{i \in I} D_i r_{ij} \right) C2_{jj'} s_{jj'} + \sum_{j \in J} F1_j x_j - \sum_{k \in K} \lambda_k V_k \\ & + \sum_{j' \in J} \sum_{k \in K} \left(\sum_{j \in J} \left(\sum_{i \in I} D_i r_{ij} \right) s_{jj'} \right) (C3_{j'k} + \lambda_k \rho_k) t_{j'k}. \end{aligned} \quad (2.18)$$

The last term of Equation 2.18 implies that recosting the transport cost $C3_{j'k}$ can conveniently play the role of coordination. It is simply carried out as $C3_{j'k} := C3_{j'k} + \text{constant} \times \rho_k$. From the statements so far, we know that the coordination can be viewed as the auction on the transportation cost so that the procurement becomes most suitable for the entire chain. By virtue of the increase in accuracy by computing V_k and R_k along with the iteration, we can expect convergence from such a coordination.

(3) Procedure for a Coordinated Solution

To reduce the computation load, we further break down each sub-problem into two levels, *i.e.*, the upper level problem to determine the locations and the lower one to determine the routes. Taking such hierarchical approach, we can apply such a hybrid method that will bring about the following advantages:

- In the upper level problem, we can shrink the search space dramatically by confining the search to location only.
- The lower level problem is transformed into a problem that is possible to solve extremely effectively.

As a drawback, we need to solve repeatedly one of the two sub-problems subject to the foregoing result of the other sub-problem in turn. However, the computational load of such an adjustment is revealed to be moderate and effective [27].

Presently, we can solve the upstream problem following the method that applies tabu search [9, 10] for the upper level and mathematical programming for the lower level (hybrid tabu search). Moreover, the lower problem of the upstream network becomes a special type of linear programming referring to the minimum cost flow (MCF) problem [42]. In practice, the original graph representing physical flow (Figure 2.16a) can be transformed into the graph shown in Figure 2.16b, where the label on an arrow and edge indicate cost and capacity, respectively. This transformation is carried out based on the following procedure.

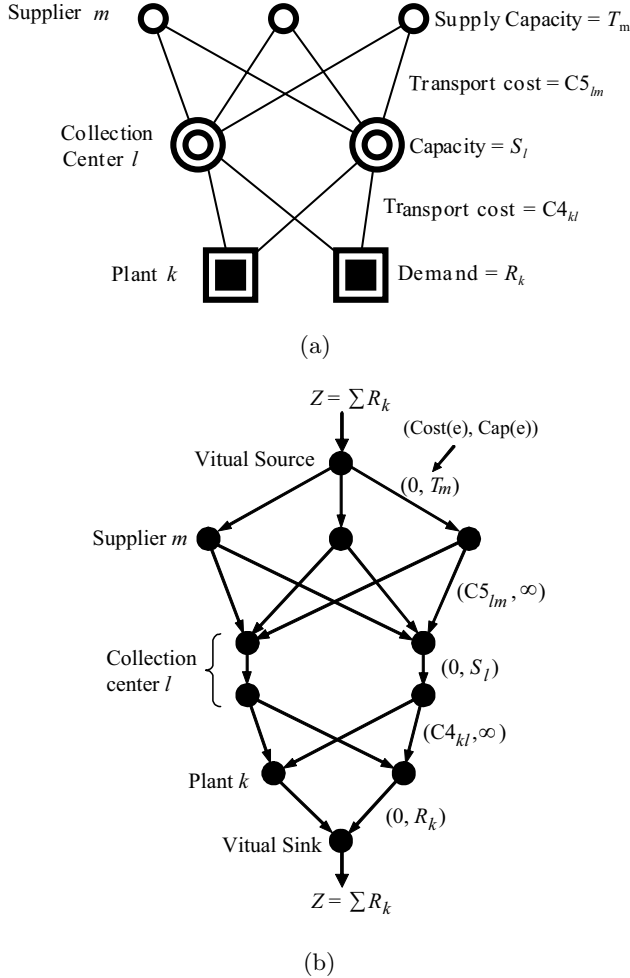


Fig. 2.16. Transformation of the flow graph: (a) physical flow graph, (b) minimum cost flow graph

- Step 1: Place the node corresponding to each facility. In particular, double the nodes of hub facilities (CC).
- Step 2: Add two imaginary nodes termed source (root of the graph) at the top of the graph and the node termed sink at the bottom of the graph.
- Step 3: Connect between nodes with the edge labeled by $(cost(e), capacity(e))$ as follows:
- label the edge between source and supplier by $(0, T_m)$,
 - label the edge between supplier and CC by $(C5_{lm}, \infty)$,
 - label the edge between the duplicated CC by $(0, S_l)$,
 - label the edge between CC and plant by $(C4_{kl}, \infty)$,
 - label the edge between plant and sink by $(0, D_k)$.
- Step 4: Set the amount of flow $\Sigma_i D_i$ at the source so that the total demand is satisfied.

On the other hand, in the downstream problem, the lower level problem refers to the IP due to the single allocation condition. It is described as the shortest path problem if we neglect the capacity constraints on DCs or Equations 2.10 and 2.12. After all, it is possible to provide another efficient hybrid tabu search that employs the sophisticated Dijkstra method to solve the shortest path problem with the capacity constraints [43]. First, the Lagrange relaxation is used to cope with the capacity constraints. Then the idea simulating an auction on the transport cost is conveniently applied. Thereat, if a certain DC would not satisfy its capacity constraint, we can consider that it occurred due to the too cheap transport costs connectable to that DC. So if we raise such cost, some connections may move on other cheaper routes in the next call. Thus adjusting the transportation cost depending on the violation amount like $\hat{C}1_{ij} := C1_{ij} + \mu \cdot \Delta P_i$, and similarly for C2, all constraints are expected to be satisfied at last. Here μ and ΔP_i denote a coefficient related to Lagrange multiplier and the violated amount at the i -th DC.

Finally, the entire procedure is summarized as follows.

- Step 1: Set all parameters at their initial values.
- Step 2: Under the prescribed parameters, solve the downstream problem by using hybrid tabu search.
- 2.1: Provide the initial location of DCs.
 - 2.2: Decide on the routes covering the plants, DCs, and customers by solving the capacitated shortest path problem.
 - 2.3: Revise the DCs' location repeatedly until the stopping condition of tabu search is satisfied.
- Step 3: Compute the necessary amount of the plant based on the above result.
- Step 4: Solve the upstream problem using hybrid tabu search.
- 4.1: Provide the initial location of CCs.
 - 4.2: Decide on the routes covering the suppliers, CCs and plants from MCF problem.
 - 4.3: Revise the CCs' location according to tabu search.

Step 5: Check the stopping condition. If it is satisfied, stop.

Step 6: Recalculate the transport costs between plants and DCs, and go back to Step 2.

B. Example of Supply Chain Optimization

The performance of the above method is evaluated by solving a variety of benchmark problems whose features are summarized in Table 2.1. They are produced by generating the nodes whose appearance rates become approximately 3: 4: 1: 6: 8 among suppliers, CCs plants, DCs, and customers. Then the transport cost per unit demand is given by the value corresponding to the Euclid distance between each node. The demand and capacity are decided randomly between certain intervals.

Table 2.1. Properties of the benchmark problem

Prob. ID	Sply	CC site	Plant	DC site	Cust	Combination*
b6	84	96	6	108	120	2.6×10^{61}
b7	98	112	7	126	140	4.4×10^{71}
b8	112	128	8	144	160	7.6×10^{81}

* Number of combinations regarding CC and DC locations

In tabu search, we explore the local search space by applying three operations such as add, subtract, and swap with the prescribed probability as shown in Table 2.2. By letting the attributes of the candidates for neighbor state be open and closed, we provide the following two rules to prepare a tabu list with a length of 50.

Rule 1: Prohibit the exchange of attributes when the updated solution can improve the current solution.

Rule 2: Prohibit keeping the attribute as it is when the updated solution fails.

Table 2.2. Employed neighborhood operations

Type	Probability	Operation
Add	$p_{\text{add}} = 0.1$	Let closed hub v_{ins} open
Subtract	$p_{\text{subtract}} = 0.5$	Let opened hub v_{del} close
Swap	$p_{\text{swap}} = 0.4$	Let closed hub v_{ins} open and opened hub v_{del} close

The results summarized in Table 2.3 reveal that the expansion of the computation load of the hybrid tabu search⁴ is considerably slow with the increase in problem size compared with commercial software like CPLEX (OPL-Studio) [45]. In Figure 2.17, we present the convergence features including those of downstream and upstream problems. Here, the coordination method works adequately to reduce the total cost by bargaining over the gain at the procurement chain for the loss at the distribution chain. In addition, only a small number of iterations (no more than ten) is required by convergence. By virtue of the generic nature of metaheuristic algorithms, this claims that the converged solution might almost attain the global optimum.

Table 2.3. Performance with commercial software

Prob. ID	Hybrid tabu search		OPL-Studio	
	Time [sec]	Appr. rate ^{*1}	Time [sec]	(rate)
b6A	123	1.005	7243	(59)
b6B	78	1.006	15018	(193)
b7A	159	1.006	27548	(173)
b7B	241	1.005	44129	(183)
b8A	231	1.006	24hr ^{*2}	(>37)
b8B	376	1.003	24hr ^{*2}	(>230)

CPU: 1GHz (Pentium3), RAM: 256MB

^{*1} Approximation rate = attained / final sol. of OPL.

^{*2} Solution by 24hr computation

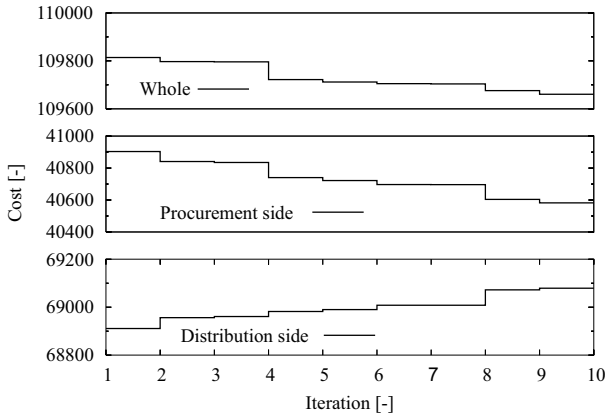


Fig. 2.17. Convergence features along the iteration

⁴ The MCF problem was solved using a code by Goldberg termed CS2 [44].

2.4.2 Sequencing Planning for a Mixed-model Assembly Line Using SA

For a relevant injection sequencing on a mixed-model assembly line, one of the major aspects is to level out the workload at each workstation against variations of assembly time per product model [46]. Another one is to keep the usage rate of every part constant at the assembly line [47]. These two aspects have been widely discussed in the literature. Usually, to keep production balance and to prevent line stoppage, a large work-in-process (WIP) inventory is required between two lines operated in different production manners, *e.g.*, the mixed-model assembly line and its preceding painting line in the car industry. In other words, achieving these two goals proportionally can bring about a reduction of the WIP inventory. In the following, therefore, we consider a sequencing problem that aims at minimizing the weighted sum of the line stoppage times and the idle time of workers.

A. Model of a Mixed-model Assembly Line with a Painting Line

Figure 2.18 shows a mixed-model assembly line including a painting line where each product is supplied from the foregoing body line every cycle time (CT). The painting line is composed of sub-painting, main painting and check processes. Re-painting repeats the main painting twice to correct defective products. The defective products are put in the buffer after correction. From the buffer, necessary amounts of product are taken out in order of the injection sequence at the mixed-model assembly line. It is equipped with K workstations on a conveyor moving at constant speed. At each workstation, a worker assembles the prescribed parts into the product models.

Furthermore, we assume the following conditions.

1. Paint defects occur at random.
2. The correction time of defective product varies randomly.
3. The production lead-time of the painting line is longer than that of the assembly line.

The sequencing problem under consideration is formulated as follows:

$$\min_{\pi \in \Pi} \rho_p \times B^t + \rho_a \times \sum_{t=1}^T \max_{1 \leq k \leq K} (P_k^t, A_k^t) + \rho_w \times \sum_{t=1}^T \sum_{k=1}^K W_k^t,$$

subject to

$$\sum_{i=1}^I z_i^t = 1, \quad t = 1, \dots, T, \quad (2.19)$$

$$\sum_{t=1}^T z_i^t = d_i, \quad i = 1, \dots, I, \quad (2.20)$$

where the notation is as follows.

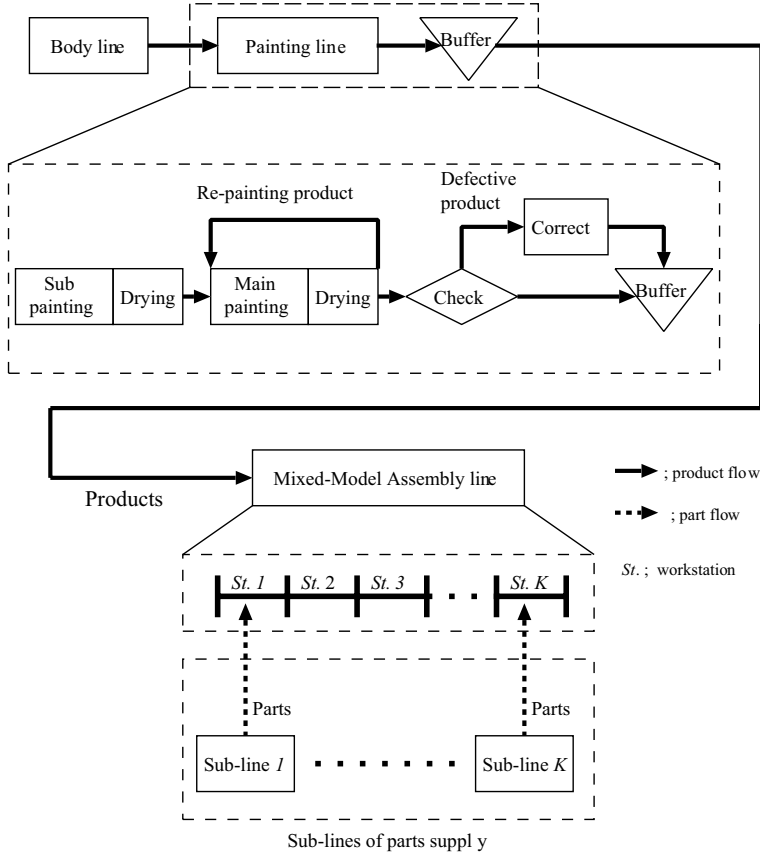


Fig. 2.18. Scheme of a mixed-model assembly line and a painting line model

I : number of product models.

K : number of workstations.

T : number of injection periods.

π : injection sequence over a planning horizon (decided from z_i^t).

Π : set of sequences ($\pi \in \Pi$).

B^t : line stoppage time due to product shortage at injection period t .

P_k^t : line stoppage time due to part shortage at workstation k at injection period t .

A_k^t : line stoppage time by work delay of a worker. This happens when the workload exceeds CT in workstation k at injection period t .

W_k^t : idle time of worker at workstation k at injection period t .

z_i^t : 0-1 variable that takes 1 if the product model i is supplied to the assembly line at injection period t . Otherwise, 0.
 d_i : demand of product model i over T .

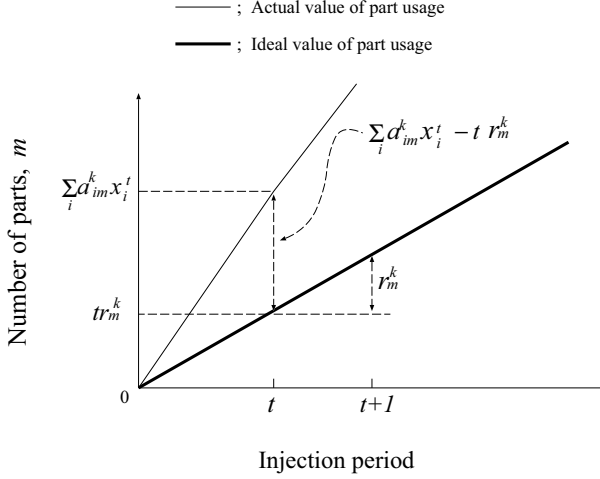


Fig. 2.19. Line stoppage time based on the goal chasing method [48]

We suppose that the objective function is described by a weighted sum of the line stoppage times and the idle time, where ρ_p , ρ_a and ρ_w are weighting factors ($0 < \rho_p, \rho_a, \rho_w < 1$). Among the constraints, Equation 2.19 indicates that plural products cannot be supplied simultaneously, and Equation 2.20 requires that the demand of each product model be satisfied.

Figure 2.19 illustrates a situation where the part shortage occurs at the workstation k when the quantity of part m used ($\sum_i a_{im}^k x_i^t$) exceeds its ideal quantity (tr_m^k) at the injection period t . Then, P_k^t is given as follows:

$$P_k^t = \max\left[\max_{1 \leq m \leq M} \left(\frac{\sum_{i=1}^I a_{im}^k x_i^t - tr_m^k}{r_m^k} CT\right), 0\right],$$

where a_{im}^k is the quantity of part m required for model i , x_i^t the accumulative amount of production for model i during injection period from 1 to t , *i.e.*,

$$x_i^t = \sum_{l=1}^t z_i^l, \quad (i = 1, \dots, I). \quad (2.21)$$

Moreover, r_m^k denotes the ideal usage rate of part m , and M the maximum number of parts used on the workstation.

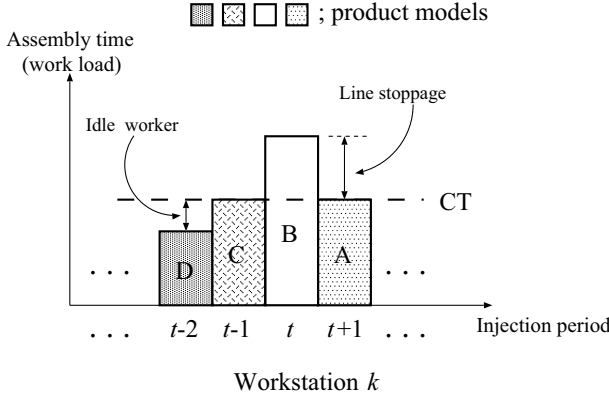


Fig. 2.20. Line stoppage due to workload unbalance

On the other hand, Figure 2.20 show a simple example of how line stoppage or idle work occurs due to variations of workloads. Each product model with different workloads are put into workstation k along injection period. Since the assembly time (workload) exceeds CT at injection period t , the line stoppage occurs whereas idle work occurs at $t - 2$. By knowing these, the line stoppage time A_k^t and the idle time W_k^t can be calculated from Equations 2.22 and Equation 2.23, respectively,

$$A_k^t = \max(L_k^t - CT, 0), \quad (2.22)$$

$$W_k^t = \max(CT - L_k^t, 0), \quad (2.23)$$

where L_k^t denotes the working time of a worker at workstation k at injection period t .

Noticing that the product models from the painting line can be viewed equivalently as the parts from a sub-line in the mixed-model assembly line, we can give the line stoppage time B^t due to part shortages as Equation 2.24,

$$B^t = \max\left(\frac{x_i^t - tr_{pi}}{r_{pi}}CT, 0\right), \quad t = 1, \dots, T, \quad i = 1, \dots, I, \quad (2.24)$$

where r_{pi} is the supply rate of product model i from the painting line over the entire injection periods. Consequently, Equation 2.24 shows the time difference between the actual injection time of the product model i and the ideal one. Here we give r_{pi} like Equation 2.25 by taking the correction time of defective products at the painting line into account,

$$r_{pi} = \frac{d_i}{T + [\sigma d_i C_i]}, \quad i = 1, \dots, I, \quad (2.25)$$

where σ is the defective rate of products at the painting line, C_i the correction time for the defective product model i , and $[\cdot]$ is a Gauss symbol.

Furthermore, to improve the above prediction, r_{pi} is revised at every production period ($n = 1, \dots, N$) according to the following procedures.

Step 1: Forecast r_{pi} from the input order to the painting line at $n = 1$ (see Figure 2.21a).

Step 2: After the injection at production period n is completed, obtain the quantity and the completion time (called “delivery-information” hereinafter) of product model i in the buffer.

Step 3: Update r_{pi} based on the delivery information of model i acquired at $n - 1$ (see Figure 2.21b).

3-1: Generate the supply rate F_{ij} ($j = 1, 2, \dots$) at every injection period when product model i is put into the buffer.

3-2: Average F_{ij} and r_{pi} to obtain the supply rate r'_{pi} of the product model i at n .

Step4: If $n = N$, stop. Otherwise, Let $n := n + 1$ and go back to Step 2.

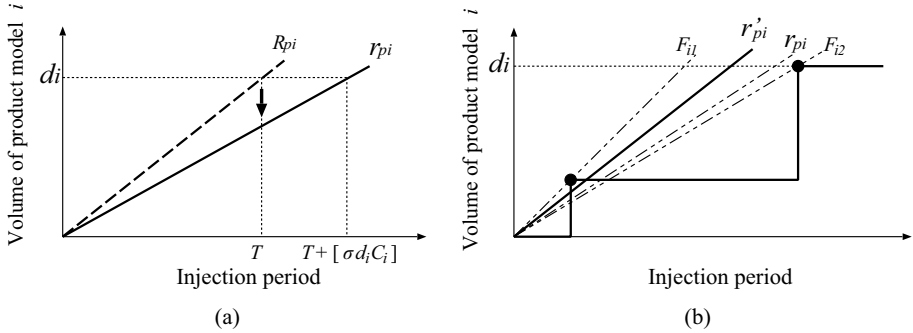


Fig. 2.21. Forecast scheme of r_{pi} and r'_{pi} : (a) estimation of r_{pi} ($n = 1$), (b) re-evaluation of r_{pi} ($n > 1$)

B. An Example of a Mixed-model Assembly Line

Numerical experiments are carried out under the conditions shown in Table 2.4. Weighting factors ρ_p , ρ_a and ρ_w are set as 0.5, 0.4 and 0.1, respectively. Moreover, the results are evaluated based on the average over 100 data sets generated randomly. To cope with the sequencing problem that belongs to a NP-hard solution procedure, SA is applied as a solution method for deriving a near optimal solution. We give a reference state by the random sequence

of injection so as to satisfy Equations 2.19 and 2.20. Then swapping two arbitrarily chosen product models in the sequence generates the neighbors of state. In the exponential cooling schedule, the temperature decreases by a fixed factor 0.8 at each step from the initial temperature 100 to the end during 150 iterations.

Table 2.4. Input parameters

Cycle time, CT [min]	5
Station number, K	100
Product model, I	10
Total production number, $\sum_i d_i$	100
Injection period, T	100
Production period, N	30
Defective rate	0.2
Correction time [min]	[15, 25]

The advantages of the total optimization (“Total sequencing”) were compared with the result obtained when neglecting the two terms in the objective function, *i.e.*, $\rho_p = \rho_w = 0$ (“Level sequencing”).

Table 2.5. Comparison of sequencing strategies

	WIP inventory volume	Line stoppage time [min]	Idle time [min]
Total sequencing	28.7	43.7	4.2
Level sequencing	37.5	31.2	4.1

In Table 2.5, the WIP inventory volume means the value necessary for preventing line stoppage due to product shortage while the line stoppage time and idle time are the times incurred by the non-leveling of the parts usage and the workloads at the assembly line, respectively. Though the WIP inventory of “Total sequencing” is smaller than that of the “Level sequencing”, the line stoppage and the idle times are a little inferior to the previous result. Therefore, the advantage of the optimization actually refers to the relevant management of the WIP inventory between two lines. As illustrated in Figure 2.22, “Total sequencing” is known to achieve the drastic decrease and stable volume in the inventory compared with “Level sequencing”.

2.4.3 General Scheduling Considering Human–Machine Cooperation

A number of resources controlled by computers are now popular in manufacturing *e.g.*, CNC machine tools, robots, AGVs, and automated warehouses.

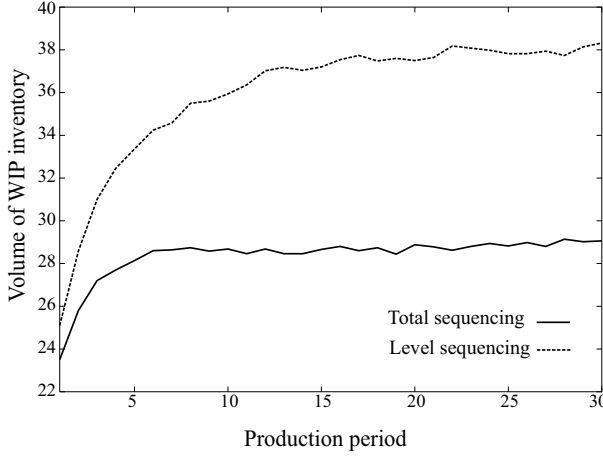


Fig. 2.22. Features of the WIP inventory along a production period

There, the role of computers is to execute the prescribed tasks automatically according to the production plans. Therefore, the advanced production resources automated by the computer are expected to explore the next generation of manufacturing systems [49]. In the near future, autonomous machine tools and robots might produce various products in flexible manners. In the current systems, however, the role of the human operator is still important. In many factories, multi-skilled operators manipulate the multiple machine tools while moving among the multiple resources. Such a situation makes it meaningless to ignore the role of operators and make a plan confined only to the status of non-human resources.

This point of view requires us to generalize the scheduling problem associated with the cooperation between human operators and resources [37]. Based on the relationship between the resources assigned to the job, incidental operations such as loading and unloading of the products are analyzed according to material flows. Then, a modified dispatching rule is applied to solve the scheduling problem.

A. Operation Classes for Generating a Schedule

The following notations will be used since production is related to a number of jobs, operations and processes associated with the job. Moreover, the term “process” will be used when we emphasize dealing with a product while “operation” will be used when we represent the manipulation of resources.

$j_{\eta,i}^{\zeta,v}$: v -th operation processed by resource ζ and i -th process for product η regarding parameter j .

s : starting time of the job.

f : finishing time of the job.
 p : processing time of the job.

The scheduling problem is usually formulated under the following assumptions.

1. Every resource can perform only one job at a time.
2. Every resource can start an operation after a preceding process has been finished.
3. The processing order and the processing time are given, and any change of the processing order is prohibited.

Under these conditions, the scheduling is to determine the operating order assigned to each resource. Figure 2.23 illustrates the Gantt charts for two possible situations of a job processed by machines ξ and ζ . As shown in Figure 2.23a, it is possible to start the target operation of resource ζ immediately after the preceding operation has been finished. In contrast, as shown in Figure 2.23b, since machine ζ can perform only one job at a time, resource ζ cannot begin to process even if resource ξ has finished the preceding process.

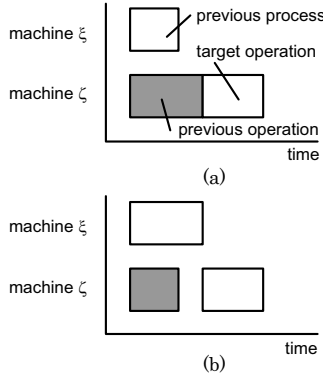


Fig. 2.23. Dependency of jobs processed by two machines: (a) on the previous operation, (b) on the previous process

Therefore, the starting time of the target job can be determined as follows:

$$s_{\eta,i}^{\zeta,v} = \max[f^{\zeta,v-1}, f_{\eta,i-1}], \quad (2.26)$$

where operator $\max[\cdot]$ returns the greatest value of the arguments. On the other hand, the finishing time is calculated by the following equation:

$$f_{\eta,i}^{\zeta,v} = s_{\eta,i}^{\zeta,v} + p_{\eta,i}^{\zeta,v}.$$

In addition, we need to consider the following aspects for the generalization of scheduling. In the conventional scheduling problem, it is assumed that each resource receives one job from another resource, then processes it and transfers it to another resource. However, in real world manufacturing, multiple resources are commonly employed to process a job. Figure 2.24 shows three types of Gantt charts for cases where multiple resources are used for manufacturing.

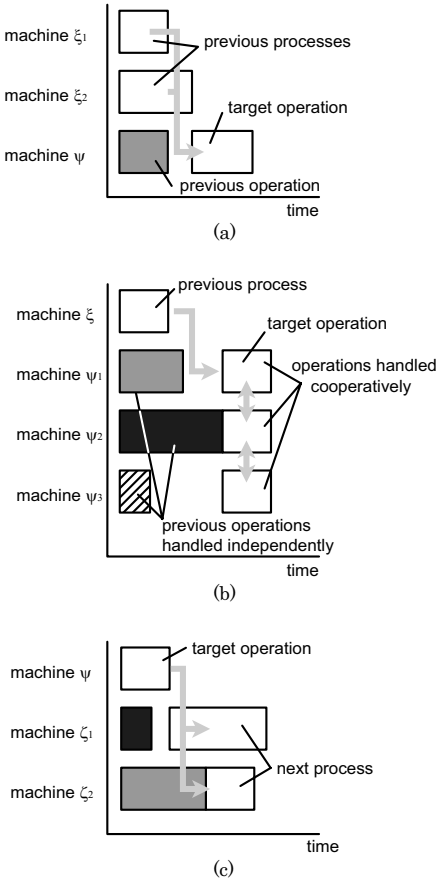


Fig. 2.24. Classification of a schedule based on material flows: (a) parts supplied from multiple machines, (b) operations handled cooperatively by multiple machines, and (c) operations of plural jobs using parts supplied from one machine

In the first case (a), one resource receives one job from the multiple resources. This type of material flow, called “merge”, corresponds to the case where a robot assembles multiple parts supplied to it from the multiple re-

sources, for example. In the second case (b), multiple resources are assigned to a job. However, each resource cannot begin to process the job until all resources have finished the preceding jobs. This type of production is known as “cooperation”. Examples of the cooperation are cases where an operator manipulates a machine tool, and where a handling robot transfers a job from AGV to machine tool. The last one (c) corresponds to “distribution”, which is the case where several resources receive the job individually from another resource. Carrying several types of parts by truck from a subcontractor is a typical example of this case. Various resources cannot begin to process until all trucks arrive at the factory. In these cases, the starting time of the target job is determined as follows.

$$s_{\eta,i}^{\psi_{\alpha},v} = \max[\{f_{\eta,i-1}^{\xi_{\gamma},v-1}\}, \{f^{\psi_{\beta},w-1}\}, f^{\psi_{\alpha},v-1}],$$

where ξ_{γ} is every resource processing the preceding process of the job $j_{\eta,i}^{\psi_{\alpha},v}$ and ψ_{β} every resource processing the job cooperatively with resource ψ_{α} . Resource ψ_{β} processes the job $j_{\eta,i-1}$ as the w -th operation, and $\{\cdot\}$ shows a set of finishing times f .

Jobs like loading and unloading are respectively considered as a pre-operation and a post-operation incidental to the main job (incidental operation). Status check and execution of NC program by a human operator are also viewed as such operations. In conventional scheduling, these jobs are likely to be ignored because they take a much shorter time compared with the main job. However, the role of these operations are still essential whenever their processed times are insignificant. For example, the resources cannot begin the process without a safety check by a human operator even in current automated manufacturing.

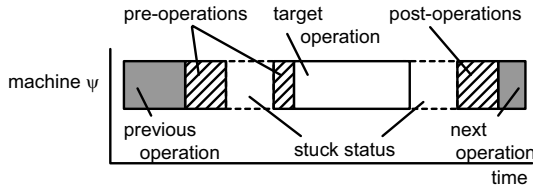


Fig. 2.25. Pre-operation and post-operation

Figure 2.25 illustrates the case where multiple pre-operations and post-operations are related to the main job (noted as the target operation). Between the two incidental operations and/or between the incidental operation and the main job, there arises an undesirable idle time or stuck time during which the resource cannot execute the other job. For generalizing the scheduling, concerns with these operations are also unavoidable.

B. Solution Method

Generally speaking, an appropriate dispatching rule can derive a practical schedule even for the real world problem with a large number of products and resources. To deal with the complicated situations mentioned above in a practical manner, it makes sense to apply this kind of knowledge or an empirical optimization method. A modified earliest start time (EST) rule is effective for obtaining a schedule to level out the waiting times. It is employed as follows.

- Step 1: Make an executable job list $\{j_{\eta,i}^{\zeta,v}\}$ where job $j_{\eta,i}^{\zeta,v}$ is the first job of the product or the preceding job $j_{\eta,i-1}$ assigned on the schedule.
- Step 2: Calculate the starting time $s_{\eta,i}^{\zeta,v}$ of the job $j_{\eta,i}^{\zeta,v}$ by Equation 2.26. If the operator manipulating machine ζ for processing job $j_{\eta,i}^{\zeta,v}$ engaged in the manipulation of another machine ξ before $j_{\eta,i}^{\zeta,v}$, then modify $s_{\eta,i}^{\zeta,v}$ using the following equation:

$$\hat{s}_{\eta,i}^{\zeta,v} = s_{\eta,i}^{\zeta,v} + t_{\xi,\zeta},$$

where $t_{\xi,\zeta}$ is the moving time of the operator from machine ξ to machine ζ .

- Step 3: Select the job that can begin the process earliest. If there are plural candidates, select the job that has the most work to do.
- Step 4: Repeat from Step 1 through 3 until all jobs are assigned to the resources.

C. Examples of a Schedule with a Human Operator

To illustrate the validity of the above discussions, a job shop scheduling problem is solved under the following conditions. Two multi-skilled operators and eight machine tools produce ten products. Both operators can manipulate multiple machine tools. Every job processed by the machine tool requires pre-operation and post-operation by the human operators. These incidental jobs are also identified as the jobs that need cooperation between human operators and machines.

Figure 2.26 shows a Gantt chart partially extracted from the scheduling obtained here. As shown in these figures, one operator manipulates the machine both at the beginning and at the end of jobs. Figure 2.26b shows the case where the moving time of an operator between two machines is short and the operator can move to machine ζ immediately after loading on machine ξ . Staying at machine ζ until the unloading of job B, the operator can return to machine ξ without any delay for unloading job A.

On the other hand, Figure 2.26c shows the case where the operator takes double time to move between these two machines. However, the operating order is the same as before, the stuck time occurs on machine ξ due to the late arrival of the operator.

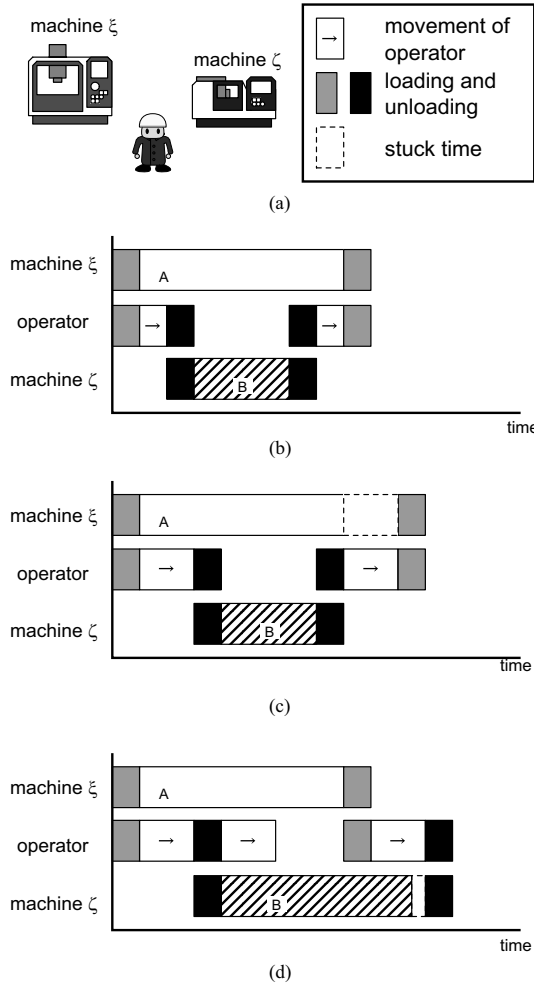


Fig. 2.26. Examples of scheduling with a human operator: (a) operator and machine tools, (b) schedule with loading and unloading by an operator, (c) schedule when an operator takes double time for movement between machine tools, and (d) schedule when job B takes double time for operation

Moreover, Figure 2.26d shows the influence of the job processing time. If the processing time of job B is double, it wastes much time because the operator will not stay at machine ζ . The operator returns to machine ξ immediately after setting up the job on machine ζ and waits for job A to be completed by machine ξ . The stuck time occurring on machine ζ becomes shorter compared with the stuck time occurring on machine ξ if the operator stays at machine ζ . This example clearly reveals the importance of the contribution of operators for a practical schedule.

2.5 Optimization under Uncertainty

There exist more or less uncertain factors in mathematical models employed for manufacturing optimization. As the lead-time for system development, planning and design become longer, systems will suffer unexpected deviations more often and more seriously. However, since it is impossible to forecast every unknown or uncertain factor beforehand, we need to analyze in advance the influence of such uncertainties on state and performance before optimization. Without considering various uncertainties involved in the system model, it may happen that the optimum solution is useful only in the specific situation, or at worst becomes insignificant. Especially when engaging in the real world problems, such an understanding is of special importance to guarantee a certain security, confidence, and economical merit.

There are known several types of uncertainty, associated with the optimization problems, *i.e.*, parameter deviations involved in the objective function and constraints; structural errors of the system model, *e.g.*, linear/non-linear, missing/redundant variables and/or constraints, *etc.* Regarding the nature of uncertain parameters, they are also classified into categories, *i.e.*, deterministic, stochastic and fuzzy deviations. To cope with the uncertainties associated with the optimization problem either explicitly or inexplicitly, much research has been carried out for many years. They refer to technical terms such as sensitivity, flexibility, robustness, and so on. Stochastic optimization, chance constrained optimization and fuzzy optimization are popularly known classes of optimization problems associated with uncertainties.

Leaving the introduction of these approaches to other literature [50], a new interest related to the recent development of metaheuristic optimization methods will be considered here. Deriving an insensitive solution against uncertainties is a major interest in this section.

2.5.1 A GA to Derive an Insensitive Solution against Uncertain Parameters

It is desirable to make the optimal solution adapt dynamically according to the deviation of parameters and/or changes of the environment. For various reasons, however, such a dynamic adaptability is not easy to achieve. Instead, we might take a proper precaution and try to obtain a solution that is robust against the changes. For this purpose, such a problem is often formulated as a stochastic optimization problem that will maximize the expectation of the objective function with uncertain parameters. Similarly, we introduce a few GA methods where fitness is calculated by stochastic parameters like expectation and variance of the objective function. Though GA has been applied to many deterministic optimizations, not so many studies have been carried out on the uncertainties [51, 52, 53, 54]. However, by virtue of the population-based search method through natural selection, GA has a high potential ability to cope with the uncertainties.

First, let us consider the deterministic optimization problem described as follows:

$$[Problem] \quad \min f(x) \quad \text{subject to} \quad x \in X \subseteq \mathbb{R}^n,$$

where x denotes a decision variable vector and X its admissible region. Moreover, f is an objective function. On the other hand, the optimization problem under uncertainty is given by

$$[Problem] \quad \min F_w(f(x, w)) \quad \text{subject to} \quad \begin{cases} x \in X \subseteq \mathbb{R}^n \\ w \in W \subseteq \mathbb{R}^m \end{cases}.$$

Since GA popularly handles constraints with the penalty function method, below the uncertainties are assumed to be involved only in the objective function without loss of generality. Moreover, if the influence from uncertainties is evaluated through expectation, the above problem can be re-described as follows:

$$[Problem] \quad \min E_w[f(x, w)] \quad \text{subject to} \quad x \in X \subseteq \mathbb{R}^n,$$

where $E_w[\cdot]$ denotes the expectation with respect to w . When the probabilistic distribution function $\varphi(w)$ is given, it is calculated by the following equation:

$$E_w = \int_{-\infty}^{\infty} \varphi(w) f(x, w) dw.$$

On the other hand, when the uncertain parameters deviate randomly within a certain interval, or the probabilistic distribution function is not given explicitly, the above computation is substituted by the average over K samples. In this case, a large number of samples can increase the accuracy of such a computation,

$$E_w = \frac{1}{K} \sum_{i=1}^K f(x, w_i).$$

Due to the generic property compared to the natural selection, in GA, individuals with higher adaptability can survive to the next generation even in an environment suffering from (parameter) deviations. This means that these survivors have been exposed to various parameter deviations during all generations long. Accordingly, the solutions obtained there are to be selected based on the expectation computed through a large number of sampling eventually or the most precise evaluation. In other words, GA can concern the uncertain problem altogether and all over the generation as well. Noting the high computational load of GA, however, how to reduce the additional load consumed for such a computation becomes a major point in developing effective methods.

The first method applies the usual GA by simply calculating the fitness from the expectation in terms of the sufficient number of samples in every generation, *i.e.*, $F_i = E_w[\cdot]$. As easily supposed, a very large number of samples is to be evaluated by the end of the search. Usually, the same stopping condition is adopted as same as in the usual GA.

Since the dominant individuals are to be evaluated repeatedly over the generation, it is possible to reduce the load necessary for the correct evaluation of expectation if the inherited information is available. Based on such prospects, the second method [49] uses Equation 2.27 for the calculation of fitness (for simplicity, the following equations are described assuming decision variable is scalar):

$$F_i = \frac{Age_i - 1)H(P_i) + f(x_i, w_j)}{Age_i}, \quad (2.27)$$

where F_i is the fitness of the i -th chromosome, $H(P_i)$ the fitness of one of the parent being closer to each offspring in the search space (its distance is denoted by D). Age_i corresponds to the individual's age that increases with the generation by one, but is reset every generation with the probability $1 - p(D)$. Here, $p(D)$ is given as

$$p(D) = \exp\left(-\frac{D^2}{\alpha}\right),$$

where α is a constant adjusting the degree of inheritance. As α becomes larger, it is more likely to inherit the character from the parent and *vice versa*. Since the sampling is limited to only one, this method weighs the contribution of the inheritance based on insufficient information too much on the evaluation of fitness. The individual with the highest age is chosen as the converged solution.

To compromise the foregoing two methods, the third method [56] illustrated in Figure 2.27 takes multiple samplings that are not so large but not only one. They are used to calculate not only the expectation but also the variance. The additional information from the variance can compensate the insufficiency of the inherited information available at the present generation in Equation 2.27. Eventually, the fitness of the i -th individual is given by the following equation:

$$F_i = \frac{(Age_i - 1)H(P_i) - h(\bar{f}_i, \sigma_i^2)}{Age_i},$$

where $h(\bar{f}_i, \sigma_i^2)$ is given by

$$h(\bar{f}_i, \sigma_i^2) = \lambda \bar{f}_i + (1 - \lambda) \sigma_i^2,$$

where λ is a weighting factor and \bar{f}_i and σ_i^2 denote the values of average and standard deviations, respectively,

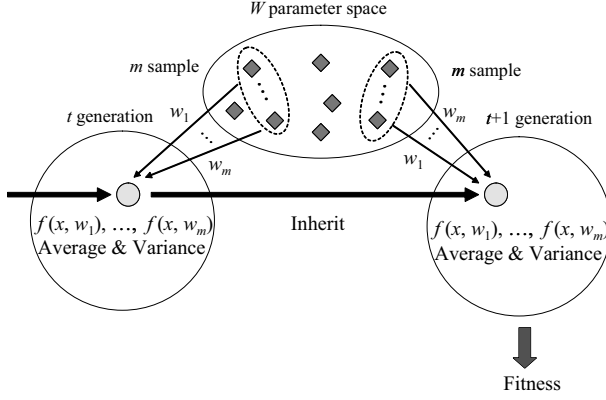


Fig. 2.27. Computation method of fitness by method 3

$$\bar{f}_i = \frac{1}{m} \sum_{j=1}^m f(x_i, w_j),$$

$$\sigma_i^2 = \frac{1}{m-1} \sum_{j=1}^m (f(x_i, w_j) - \bar{f}_i)^2,$$

where m is the sampling number.

After the stopping condition has been satisfied, the individual with the highest age is chosen as the final solution.

The first test problem to examine the performance of each method is given by the maximization of a two-peaked objective function shown in Figure 2.28.

$$f_1(x, w) = \begin{cases} A_L \sin\{B_L(x + w)\}, & (x + w \in D_L) \\ A_R \sin\{B_R(x + w - \frac{1}{11})\}, & (x + w \in D_R) \end{cases},$$

where $D_L = \{x | 0 \leq x \leq 1/11\}$, $D_R = \{x | \frac{1}{11} \leq x \leq 1\}$. A noisy parameter w deviates in two ways:

1. randomly within $[-0.004, 0.004]$
2. under the normal distribution $N[0, \sigma^2]$.

Furthermore, in the second case, two sizes of deviation are considered, *i.e.*, $\sigma = 0.01$, and 0.05 . As known from Figure 2.28, the optimal solution for each σ becomes $x_L = 0.046$ and $x_R = 0.546$, respectively. Table 2.6 compares the results obtained under the condition that the population size = 100, crossover the rate = 0.6, and the mutation rate = 0.02. After the same prescribed computation time (30 s), the final solution is chosen according to the stopping condition of each method.

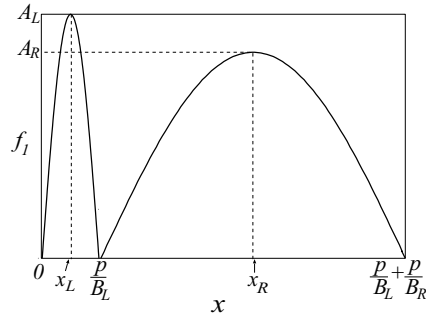


Fig. 2.28. Two-peak problem $f_1(x, w)$, ($A_L = 10$, $A_R = 8$, $B_L = 11\pi$, $B_R = 11\pi/10$, $w = 0$)

Table 2.6. Comparison of numerical results

σ	Method	Solution	Error (%)	m	Generation
0.01 ($x_L = 0.046$)	1	0.0436	4.2	20	3000
	2	0.0486	22.2	1	12000
	3	0.0458	2.3	5	8000
0.05 ($x_R = 0.546$)	1	0.539	2.0	20	3000
	2	0.523	9.1	1	12000
	3	0.545	1.7	5	8000

In every case, the third method outperforms the others. On the other hand, all results of the case $\sigma = 0.01$ are inferior to those of $\sigma = 0.05$, since around the optimal solution for $\sigma = 0.01$ (x_L), the sensitivity of f_1 with w is higher than that of the optimal solution for $\sigma = 0.05$ (x_R).

Another test problem with the five-modal objective function shown in Figure 2.29 is also solved by each method,

$$f_2(x, w) = \begin{cases} a(x, w) |\sin(5\pi(x + w))|^{0.5}, & (0.4 < x + w \leq 0.6) \\ a(x, w) \sin^6(5\pi(x + w)), & \text{otherwise} \end{cases},$$

where $a(x, w) = \exp[-2 \ln 2 (\frac{(x+w)-0.1}{0.8})^{0.2}]$.

In this problem, the noisy parameter deviates under the normal distribution with $\sigma = 0.02$ and 0.04 . As shown in Figure 2.29, the optimal solution for each deviation locates at $x_L = 0.1$ and at $x_R = 0.492$, respectively. Figure 2.30 shows the behavior of the tentative solution during the generation for $\sigma = 0.02$. From this, it is known that the third method attains the optimal solution x_L fast, and keeps it steadily. This means that the result will not be affected by the wrong selection of the stopping condition, or the oldest individual can dwell on the optimal state safely. On the other hand, the second method is inferior to the others. Figure 2.31 describes the result for $\sigma = 0.04$.

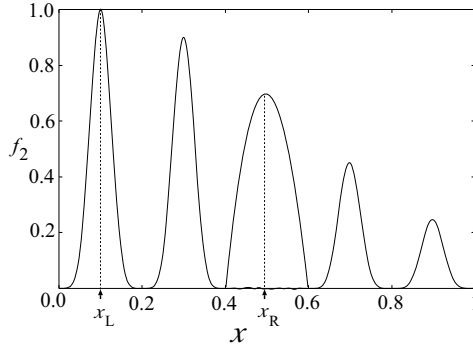


Fig. 2.29. Five-peak problem $f_2(x, w)$, ($w = 0$)

In this case, the third method also outperforms the others. These results claim that the third method can derive the solution steadily and safely regardless of the stopping conditions.

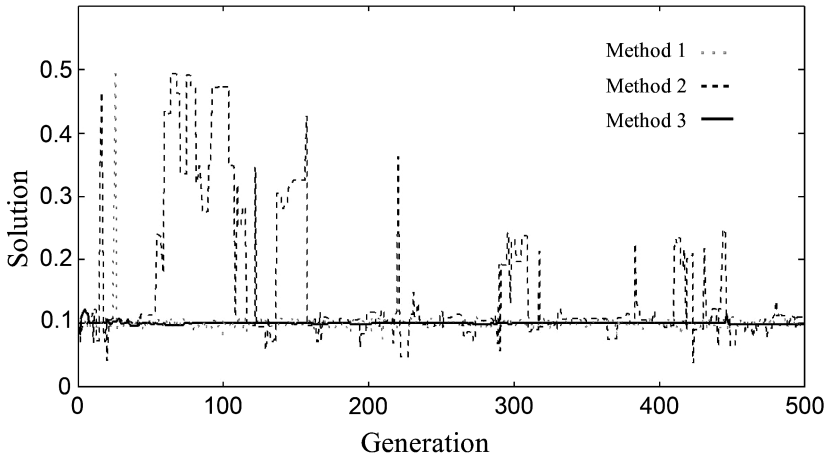


Fig. 2.30. Convergence property ($\sigma = 0.02$)

2.5.2 Flexible Logistic Network Design Optimization

Under the influence of globalization and the introduction of advanced transportation systems, industrial markets are acknowledging the importance of flexible logistic systems favoring just-in-time and agile manufacturing. Focusing on the logistic systems associated with supply chain management (SCM), a method termed hybrid tabu search is applied to solve the problem under deterministic customer demand [43]. In reality, however, a precise forecast

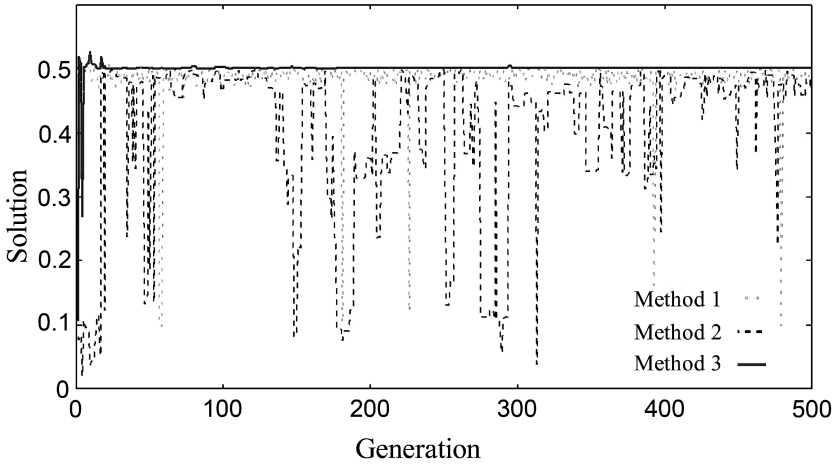


Fig. 2.31. Convergence property ($\sigma = 0.04$)

of demand is quite difficult. An incorrect estimate causes either insufficient production when forecast goes below the actual demand or undue expenditure due to large inventory. It is important, therefore, to formulate the problem by taking into account uncertainty in the demand. In fact, by assuming certain stochastic deviation, two-stage formulations using stochastic programming have been studied [57, 58]. However, these approaches seem to be ineffective for designing a flexible logistic network for the following two reasons. First, customer satisfaction is evaluated by the demand basis but it is left unrelated to other important factors like cost, flexibility, *etc.* Second, they are unconscious of taking a property of decision variables into account whether they are soft (control) or hard (design) variables.

To show an approach for deriving a flexible network against uncertain demands, let us consider a hierarchical logistic network as depicted in Figure 2.32, and define index sets I , J and K for customer (CT), distribution center (DC) and plant (PL), respectively. It is assumed that customer i has an uncertain demand D_i obeying a normal distribution. To consider this problem, a fill rate of demand termed service level is defined as follows:

$$s(\alpha\sigma) = \int_{-\infty}^{\alpha\sigma} N[p_0, \sigma] dp \quad (\alpha : \text{natural number}), \quad (2.28)$$

where $N[\cdot]$ stands for the normal distribution with average p_0 and standard deviation σ . The service level corresponds to the probability that the network can deliver products to customers whatever deviation of the demand might occur within the prescribed extent.

For example, the network designed for the average demand can present 50% service level, and 84.13% for the demand corresponding to $p_0 + \sigma$. Now the problem is to minimize the total transportation cost with respect to the lo-

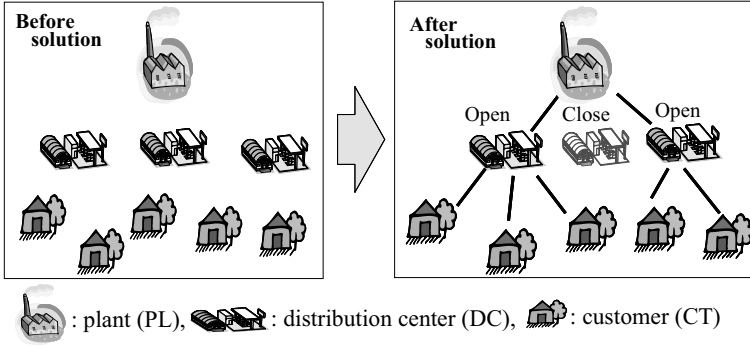


Fig. 2.32. Scheme of a logistic network

cation of DC and the selection of a route between the facilities while satisfying the service level. The following development also assumes the following:

1. Every customer is supplied via a route only as from PL to DC and from DC to CT.
2. To avoid a separate delivery, each connection is limited to only one linkage (single allocation).

Now, the problem without taking the demand deviation into account is given by the following mixed 0-1 programs [40], which is a variant formulation⁵ of the downstream problem of logistic optimization in Sect. 2.4.1:

$$[Problem] \quad \min \sum_i \sum_j f_{ij} E_{ij} + \sum_j \sum_k g_{jk} G_{jk}, \quad (2.29)$$

subject to

$$\sum_j y_{ij} = 1, \quad \forall i \in I, \quad (2.30)$$

$$f_{ij} \geq y_{ij} D_i, \quad \forall i \in I, \quad \forall j \in J, \quad (2.31)$$

$$\sum_i f_{ij} \leq x_j U_j, \quad \forall j \in J, \quad (2.32)$$

$$x_j = \sum_k z_{jk} M, \quad \forall j \in J, \quad (2.33)$$

$$g_{jk} \leq z_{jk} M, \quad \forall j \in J, \quad \forall k \in K, \quad (2.34)$$

$$\sum_k g_{jk} = \sum_{ij} f_{ij}, \quad \forall j \in J, \quad (2.35)$$

⁵ Fixed charge of location is ignored. Instead, the number of locations is set at p and delivery between DC and DC is prohibited in this model.

$$\sum_j g_{jk} \leq S_k, \quad \forall k \in K, \quad (2.36)$$

$$\sum_j x_j = p, \quad (2.37)$$

$$f, g : \text{integer}, x, y, z \in \{0, 1\},$$

where x_j denotes a binary variable that takes 1 when DC opens at the j -th candidate and 0 otherwise. The binary variables y_{ij} and z_{jk} represent the status of connection between CT and DC, and DC and PL, respectively. These two binary variables (y_{ij} and z_{jk}) become 1 when connected and 0 otherwise. Quantities f_{ij} and g_{jk} are shipping amounts from DC to CT, and from PL to DC, respectively.

The objective function stands for the total transportation cost where E_{ij} denotes unit transportation cost between the i -th CT and the j -th DC and G_{jk} that between the j -th DC and the k -th PL.

On the other hand, each constraint denotes the conditions as follows: Equation 2.31 denotes demand satisfaction where D_i represents the i -th demand; Equations 2.30 and 2.33 the single linkage conditions; Equations 2.32 and 2.36 capacity constraints where U_j is capacity at the j -th DC and S_k that at the k -th PL; Equation 2.35 flow balance; Equation 2.37 the required number of open DC. Moreover, M in Equations 2.33 and 2.34 represents a very large number.

To consider the problem, the decision variables are classified into hard and soft variables depending on their generic natures. Hard variables are not allowed to change once they have been determined (*e.g.*, DC location). On the other hand, soft variables can change according to the demand deviation (*e.g.*, distribution route). Then a two-level problem is formulated based on the considerations from flexibility analysis [60] as follows:

$$[Problem] \quad \min_{x,u,w} C_T(x, u, w|p_0),$$

subject to

$$(x, u, w) \in F(x, u, w|p_0), \quad (2.38)$$

$$\|u - v\| \leq 2\xi, \quad (2.39)$$

$$\min_{x,v,w'} C_T(x, v, w'|p_r),$$

$$\text{subject to } (x, v, w') \in F(x, v, w'|p_r), \quad (2.40)$$

$$x, u, v \in \{0, 1\}, \quad w, w' : \text{integer},$$

where x denotes the location of DC (hard variable), u and v correspond to the soft variables denoting the route for the nominal (average) demands, and the deviated demands, respectively. When $\|\cdot\|$ denote the Hamming distance, ξ refers to the allowable number of route changes. This is equivalently described

as Equation 2.39. Moreover, w and w' represent the other variables in the original problem at the nominal and the deviated states, respectively.

Also, $C_T(\cdot|p_0)$ and $F(\cdot|p_0)$ in Equation 2.38 symbolically express the objective function (Equation 2.29) and the feasible region at the nominal (Equations 2.30 through 2.37), respectively. Similarly, Problem 2.40 stands for the optimization at the deviated state. Due to the linearity of the constraints regarding demand satisfaction, *i.e.*, Equation 2.31, we can easily describe the permanently feasible region [61, 62]. This condition guarantees the feasibility even in the worst case of parameter deviations regardless of the design and control adopted. Accordingly, the demand D_i in $F(\cdot|p_r)$ must be replaced with the value corresponding to the prescribed service level. Finally, the lower level problem tries to search the optimal route while satisfying the feasibility against every deviation under the DC location decided at the upper level problem.

Even in the case where uncertainties are not considered, the formulated problem belongs to the class of NP-hard problems. It becomes especially difficult to obtain a rigid optimal solution mathematically as the problem size expands. The hybrid tabu search is applied as a core method to solve this problem repeatedly for a parametric study regarding ξ . It is necessary to engage in a tradeoff analysis on the flexible logistics decision at the next stage.

The effectiveness of the approach is examined through a variety of problems where the number of customers ranges from 50 to 150. Moreover, the number of plants $|K|$, candidate DC $|J|$, designated open DC p and customer $|I|$ are set at the ratio 5: 15 : 7: 50, and these facilities are located randomly. Then unit transportation costs E_{ij} and G_{jk} are given to be proportional to the Euclid distance between them.

Three benchmark problems are solved to examine the properties of the flexible solution through comparison with other methods. Table 2.7 shows the results of the three strategies, *i.e.*, the flexible decision (F-opt.), nominal one (N-opt.), and conservative one (W-opt.). N-opt. and W-opt. are derived from the other optimizations described below, respectively,

$$\min C_T(x, u, w|p_0) \quad \text{subject to} \quad (x, u, w) \in F(x, u, w|p_0),$$

$$\min C_T(x, v, w'|p_r) \quad \text{subject to} \quad (x, v, w') \in F(x, v, w'|p_r).$$

Then, the objective values are compared with each other both at the nominal (p_o) and the worst ($p_o + 3\sigma$) states when $\xi = 5$. The values in parenthesis express the rates to the respective optimal values. In every case, N-opt. is unable to cope with the deviated state. On the other hand, though W-opt. has an advantage at the worst state, its performance degrades outstandingly at the nominal state. In contrast, F-opt. can present better performance in the nominal state while keeping a nearly optimal value in the worst case.

Results obtained from another class of problems reveal that the more difficult the decision environment and the more seriously the deviated situation become, the more the flexible design takes the advantage.

Table 2.7. Comparison of results for the benchmark problem

Problem ID (D - $ K $ - $ J (p)$ - $ I $)	Strategy	At nominal state	At worst state
D-5-15(7)-50	F-opt.	45846 (1.25)	77938 (1.04)
	N-opt.	36775 (1.00)	NA
	W-opt.	58377 (1.59)	74850 (1.00)
D-10-30(14)-100	F-opt.	38127 (1.03)	47661 (1.04)
	N-opt.	36918 (1.00)	NA
	W-opt.	39321 (1.06)	45854 (1.00)
D-15-45(21)-150	F-opt.	40886 (1.07)	48244 (1.05)
	N-opt.	38212 (1.00)	NA
	W-opt.	45834 (1.19)	45899 (1.00)

To make a final decision associated with the flexibility, the dependence of adjusting margin ξ on the system performance or total cost needs to be examined. Since certain amounts of margin (ξ) can reduce the degradation of performance (total cost) effectively, we can derive a rational decision by compromising the attainability of these factors. An example of the tradeoff analysis is shown in Figure 2.33. Due to the tradeoff between the total cost and ξ , which increases along with the amount of deviation, decision making at the next step should be addressed in terms of the discussion about the sufficient service level and/or the allowable adjusting margin together with the cost factor.

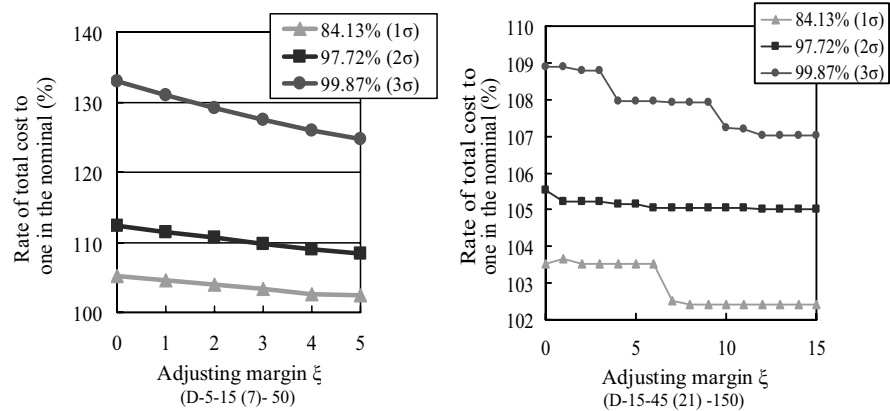


Fig. 2.33. Relation between total cost and adjusting margin ξ

2.6 Chapter Summary

In this chapter, we focused on a variety of single-objective optimization methods based on a metaheuristic approach.

These methods have emerged recently, and are nowadays filtering as practical optimization methods by virtue of the rapid progress of both computers and computer science. Roughly speaking, they are direct search methods aiming at a global optimum by utilizing a certain probabilistic drift. Their algorithms are characterized mainly by the ways in which to derive the tentative solution, how to evaluate it, and how to update it. They can even cope readily with the combinatorial optimization. Due to these favorable properties, these methods are being widely applied to some difficult problems encountered in manufacturing optimization.

To solve various complicated and large-scale problems in a numerically effective manner, we presented a hybrid approach that enables us to inherit the conventional outcomes and fuse them together with the recent outcomes straightforwardly. Types of hybrid approaches were classified, and an illustrative formulation was presented in terms of the combination of traditional mathematical programming and metaheuristic optimization in a hierarchical manner.

Then, three applications to manufacturing optimization were demonstrated to show how effectively each optimization method can solve each topic.

The first topic took a logistic problem associated with supply chain management that is closely related to the network design of hub systems such as transportation, telecommunication, *etc.* To deal with such large-scale and complex problems practically, a hybrid method was developed in a hierarchical manner. Through decomposing the problem into appropriate sub-problems, tabu search and the graph algorithm as a LP solver of the special class were applied to the resulting problems.

To increase the efficiency of the mixed-model assembly line for the small-lot-multi-kinds production, it is essential to prevent line stoppages incurred due to unexpected inconsistencies. The second topic concerned an injection sequencing problem under uncertainty associated with defective products. After formulating the problem, simulated annealing (SA) was employed to solve the resulting problem in a numerically effective manner.

Effective scheduling is one of the most important activities in intelligent manufacturing. However, little research has taken into account the role of human operators and cooperation between operators and resources. The third topic concerned production scheduling involving multi-skilled human operators manipulating multiple types of resources such as machine tools, robots and so on. A scheduling problem associated with human tasks was formulated and solved by an empirical optimization method known as the dispatching rule.

In the mathematical model employed for manufacturing optimization, there exist more or less uncertain factors that are impossible to forecast before-

hand. In the last section, as a new interest related to the recent development of metaheuristic optimization methods, the application of GA to derive an insensitive solution against uncertain parameters was introduced. By virtue of its generic nature as a population-based algorithm, a high potential ability of coping with the uncertainty was examined through numerical experiments.

Then, focusing on the logistic systems associated with supply chain management, the hybrid tabu search was used again to solve the problem under uncertain customer demand. The idea from flexibility analysis was applied by classifying the decision variables as to whether they are soft (control) or hard (design). The results obtained there revealed that the approach is very promising for making a flexible logistic decision under uncertainties from comprehensive points of view.

References

1. Glover F W, Kochenberger GA (2003) Handbook of metaheuristics- variable neighborhood search (international series in operations research and management science 57). Springer, Netherlands
2. Ribeiro CC, Hansen P (eds.) (2002) Essays and surveys in metaheuristics. Kluwer, Norwell
3. Chambers LD (ed.) (1999) Practical handbook of genetic algorithms: complex coding systems. CRC Press, Boca Raton
4. Davis L (1991) Handbook of genetic algorithms. Van Nostrand Reinhold, New York
5. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Kluwer, Boston
6. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor
7. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. *Science*, 220:671–680
8. Cerny V (1985) A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51
9. Glover F (1989) Tabu search: Part I. *ORSA Journal on Computing*, 1:190–206
10. Glover F (1990) Tabu search: Part II. *ORSA Journal on Computing*, 2:4–32
11. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359
12. Kennedy J, Eberhart R (1995) Particle swarm optimization. *Proc. IEEE International Conference on Neural Networks*, pp. 1942–1948
13. Reynolds CW (1987) Flocks, herds, and schools: a distributed behavioral model, in computer graphics. *Proc. SIGGRAPH '87*, vol. 4, pp. 25–34
14. Dorigo M, Maniezzo V, Colomi A (1996) Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:29–41
15. Dorigo M, Stutzle T (2004) Ant colony optimization. MIT Press, Cambridge

16. Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech Concurrent Computation Program, C3P Report 826
17. Laguna M, Marti R (2003) Scatter search: methodology and implementation in C (Operations Research/Computer Science Interfaces Series 24). Kluwer, Norwell
18. Shimizu Y, Tachinami Y (2002) Parallel computing for solving mixed-integer programs through a hybrid genetic algorithm. *Kagaku Kogaku Ronbunshu*, 28:268–272 (in Japanese)
19. Karimi IA, Srinivasan R, Han PL (2002) Unlock supply chain improvements through effective logistic. *Chemical Engineering Progress*, 98:32–38
20. Knolmayer G, Mertens P, Zeier A (2002) Supply chain management based on SAP systems: order management in manufacturing companies. Springer, New York
21. Stadtler H, Kilger C (2002) Supply chain management and advanced planning: concepts, models, software, and case studies (2nd ed.). Springer, New York
22. Campbell JF (1994) A survey of network hub location. *Studies in Locational Analysis*, 6:31–49
23. Drezner Z, Hamacher HW (2002) Facility Location: applications and theory. Springer, New York
24. Ebery J, Krishnamoorth M, Ernst A, Boland N (2000) The capacitated multiple allocation hub location problem: formulations and algorithms. *European Journal of Operational Research*, 120:614–631
25. O’Kelly M E, Miller H J (1994) The hub network design problem. *J. Transport Geography*, 21:31–40
26. Lee H, Shi Y, Nazem SM, Kang SY, Park TH, Sohn MH (2001) Multicriteria hub decision making for rural area telecommunication networks. *European Journal of Operational Research*, 133:483–495
27. Wada T, Shimizu Y (2006) A hybrid metaheuristic approach for optimal design of total supply chain network. *Transaction of ISCI* 19, 2:69–77 (in Japanese), *see also* Wada T, Shimizu Y, Yoo J-K (2005) Entire supply chain optimization in terms of hybrid in approach. *Proc. 15th ESCAPE*, Barcelona, Spain, pp. 591–1596
28. Okamura K, Yamashida H (1979) A heuristic algorithm for the assembly line model-mix sequencing problem to minimize the risk of stopping the conveyor. *International Journal of Production Research*, 17:233–247
29. Yano C A, Rachamadugu R (1991) Sequencing to minimize work overload in assembly lines with product options. *Management Science*, 37:572–586
30. Yoo J-K, Moriyama T, Shimizu Y (2005) A sequencing problem in mixed-model assembly line including a painting line. *Proc. ICCAS2005*, Gyeonggi-Do, Korea, pp. 1118–1122
31. Pinedo M (2002) Scheduling: theory, algorithms, and systems (2nd ed.). Prentice Hall, Upper Saddle River
32. Blazewicz J, Ecker KH, Pesch E, Schmidt G, Weglarz J (2001) Scheduling computer and manufacturing processes (2nd ed.). Springer, Berlin
33. Muth JF, Thompson GL (1963) Industrial scheduling. Prentice Hall, Englewood Cliffs
34. Brucker P (2001) Scheduling algorithms. Springer, New York
35. Calrier J (1982) The one-machine sequencing problem. *European Journal of Operation Research*, 11:42–47

36. Iwata K et al. (1980) Jobshop scheduling with operators and proxy machines. Transactions of JSME, 417:709–718
37. Hino R, Kobayashi Y, Yoo J-K, Shimizu Y (2004) Generalization of scheduling problem associated with cooperation among human operators and machine. Proc. Japan–USA Symposium on Flexible Automation, Denver
38. Garcia-Flores R, Wang XZ, Goltz GE (2000) Agent-based information flow process industries' supply chain modeling. Computers & Chemical Engineering, 24:1135–1141
39. Gupta A, Maranas CD, McDonald CM (2000) Mid-term supply chain planning under demand uncertainty: customer demand satisfaction and inventory management. Computers & Chemical Engineering, 24:2613–2621
40. Zhou Z, Cheng S, Hua B (2000) Supply chain optimization of continuous process industries with sustainability considerations. Computers & Chemical Engineering, 24:1151–1158
41. Wada T, Yamazaki Y, Shimizu Y (2007) Logistic optimization using hybrid metaheuristic approach—consideration on multi-commodity and volume discount. Transactions of JSME, 73:919–926 (in Japanese), *see also* Wada T, Yamazaki Y, Shimizu Y (2007) Logistic optimization using hybrid metaheuristic approach under very realistic conditions. Proc. 17th ESCAPE, Bucharest, Romania, pp. 733–738
42. Hassin R (1983) The minimum cost flow problem: a unifying approach to existing algorithms and a new tree search algorithm. Mathematical Programming, 25:228–239
43. Shimizu Y, Wada T (2004) Hybrid tabu search approach for hierarchical logistics optimization. Transactions of ISCIE 17, 6:241–248 (in Japanese), *see also* Logistic optimization for site location and route selection under capacity constraints using hybrid tabu search. Proc. 8th International Symposium on PSE, pp. 612–617
44. Goldberg: AV (1997) An efficient implementation of a scaling minimum-cost flow algorithm. Algorithms, 22:1–29
45. <http://www.ilog.co.jp>
46. Miltenburg J (1989) Level schedules for mixed-model assemble lines in just-in-time production systems. Management Science, 35:192–207
47. Duplaga E A, Bragg DJ (1998) Mixed-model assembly line sequencing heuristics for smoothing component parts usage. International Journal of Production Research, 36:2209–2224
48. Monden Y (1991) Toyota production system: an integrated approach to Just-In-Time. Chapman & Hall, London
49. Koren Y, Heisel U, Jovane F, Moriwaki T, Pritshow G, Ulsoy G, Van BH (1999) Reconfigurable manufacturing systems. Annals of the CIRP, 48:527–540
50. Ruszczyński A, Shapiro A (eds.) (2003) Stochastic programming. Elsevier, London
51. Branke J (2002) Evolutionary optimization in dynamic environments. Kluwer, Norwell
52. Fitzpatrick JM, Grefenstette JJ (1988) Genetic algorithms in noisy environments. Machine Learning, 3:101–120
53. Hughes EJ (2001) Evolutionary multi-objective ranking with uncertainty and noise. In: Zitzler E et al.(eds.) EMO 2001. Springer, Berlin, pp. 329–343

54. Sano Y, Kita H (2002) Optimization of noisy fitness functions by means of genetic algorithms using history of search. Transactions of IEE Japan, 122-C, 6:1001–1008 (in Japanese)
55. Tamaki H, Arai T, Abe S (1999) A genetic algorithm approach to optimization problems with uncertainties. Transactions of ISCIE, 12:297–303 (in Japanese)
56. Adachi M, Yamamoto K, Shimizu Y (2003) A genetic algorithm for deriving insensitive solution against uncertain parameters. Proc. 46-th JAAC Conference, FA2-04-3, pp. 736–739 (in Japanese)
57. Jung JY, Blau G, Pekny JF, Reklaitis GV, Eversdyk D (2004) A simulation based optimization approach to supply chain management under demand uncertainty. Computers & Chemical Engineering, 28:2087–2106
58. Guillen G, Mele FD, Bagajewicz MJ, Espuna A, Puigjaner L (2005) Multiobjective supply chain design under uncertainty. Chemical Engineering Science, 60:1535–1553
59. Shimizu Y, Matsuda S, Wada T (2006) A flexible design for logistic network under uncertain demands through hybrid meta-heuristic strategy. Transactions of ISCIE, 19:342–349 (in Japanese), *see also* Flexible design of logistic network against uncertain demands through hybrid meta-heuristic method. Proc. 16th ESCAPE, Garmisch Partenkirchen, Germany, pp. 2051–2056
60. Swaney RE, Grossmann IE (1985) An index for operational flexibility in chemical process design. Part 1: formulation and theory. AIChE Journal, 31:621–630
61. Shimizu Y, Takamatsu T (1987) A design method for process systems with flexibility consideration. Kagaku Kogaku Ronbunshu, 13:574–580 (in Japanese)
62. Shimizu Y (1989) Application of flexibility analysis for compromise solution in large scale linear systems. Journal of Chemical Engineering of Japan, 22:189–194

Frontiers in Computing Technologies for Manufacturing
Applications

Shimizu, Y.; Zhong, Z.; Batres, R.

2007, XII, 312 p. 178 illus., Hardcover

ISBN: 978-1-84628-954-5