An Investigation of Genetic Algorithms for Facility Layout Problems

Kazuhiro Kado

A thesis submitted in fulfilment of the requirements
for the degree of Master of Philosophy
to the
University of Edinburgh
1995

Abstract

The Facility Layout Problem (FLP) concerns minimising total traffic cost between facilities in a particular location under given conditions including facility size and traffic between each pair of them. Because of its NP-completeness, many suboptimal methods, which look for reasonably good solutions, have been suggested. Although many papers exist which compare the performance of these methods with each other, the work is limited in the following ways: benchmark tests were done only on FLPs consisting of identical facilities; most of the algorithms being compared relied on deterministic approaches.

Genetic Algorithms (GAs), which use a stochastic approach, have been used with some success for a number of NP-complete problems, typically finding good answers but not necessarily the best. However, a range of other approaches, from traditional operations research to simulated annealing, are possible. Moreover, a GA itself can be varied in many ways.

So, in this research project, not only the investigation of GA techniques but also some comparison with other types of approaches are done on FLPs including non-identical facilities. To provide a fair basis for comparison, fifteen FLPs are drawn from recent published papers. For the representation of solutions to FLPs, a method called Slicing Tree Structure (STS) is used. STSs can represent a wide range of layouts and can be expressed by Polish expressions, which are suitable for computation. Combining some GA techniques and STS usage, I investigated six types of GAs.

By comparing the performance of GAs with other algorithms including simulated annealing and quasi-Newton methods, I confirmed that the performance of GAs was generally better than that of other algorithms. From the comparison of the performance of different GAs with each other, I found that population seeding via clustering methods clearly improved the GA performance on FLPs consisting of many facilities. Regarding the investigation of GA parameters, I observed results in FLPs consistent with other GA studies in fields different from FLPs. In addition, I produced a benchmark record of many types of GAs, which may be a good reference for future research.

Declaration

This thesis has been composed by myself and it has not been submitted in any previous application for a degree. The work reported within was executed by myself, unless otherwise stated.

September 1995

Acknowledgements

I would like to express my gratitude to my supervisors, Dr Peter Ross and Mr Dave Corne. Thanks to insightful comments and suggestions, I was able to establish a suitable research topic and to make successful headway during my research. Also, I am very grateful to them for their valuable feedback on the draft versions of my thesis.

In addition to my supervisors, I am thankful to many people in the Artificial Intelligence Department of Edinburgh University for various help while I studied here. Especially, the support of the computer officers, Mr Michael Keightley, Mr Craig Strachan and Mr Neil Brown, was essential and is throughly appreciated.

Moreover, I would like to thank my wife, Teiko, and my two-year-old son, Yuji. They encouraged, cooperated, and sometimes amused me a lot. Thanks to my family, I could intensively study in an organised way and enjoy a fulfilled life in Scotland.

Finally, I would like to say thank-you to Mazda Motor Corporation which gave me the financial assistance.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Facility Layout Problems

The facility layout problem (FLP) concerns minimising total traffic cost between facilities in a particular location under given conditions including facility size and traffic between each pair of facilities. For example, FLPs can be applied to manufacturing machines in a factory [Sou93], to people traffic in a office [Ste87], and so on.

The FLP can be regarded as a problem to find the layout minimising the following function value F [KH87].

$$F = \sum_{i=1}^{M} \sum_{j=1}^{M} (traffic)_{ij} \times (distance)_{f(i)f(j)}$$

where	$(traffic)_{ij}$	$=$	the traffic between facilities i and j
	$(distance)_{kl}$	$=$	the distance between locations of k and l
	$f(i)$	$=$	the location of facility i
	M	$=$	the number of facilities.

When all facilities require the same area, we call it an identical FLP. When this is not the case, we will call it a non-identical FLP. Owing to its NP-completeness [SG76], it is impractical to search for optimal solutions. Therefore, to look for reasonably good solutions, many suboptimal methods such as CRAFT [BAV64] and MAT [EGH70] have been suggested.

In order to compare the performance of these methods with each other, some benchmark tests have been done by [NVR68], [KH87], [YP93] and so on. But the work is limited in the following ways:

- The benchmarks were done only on FLPs consisting of identical facilities.
- Most of the algorithms to be compared relied on deterministic approaches.

Since non-identical FLPs may often appear in real situations, the comparison in previous work may lack a practical point of view. And, because the deterministic approaches such as the hill-climbing method may only reach one of many local minima, the final solution might not be sufficiently good. As [GG89] mentioned, some stochastic approaches such as Genetic Algorithms (GAs) and Simulated Annealing (SA) may be generally better at NP-complete problems including FLPs.

1.2 Genetic Algorithms

The GA is a problem solving technique hinted at by the evolution theory of living creatures [Whi93]. In GAs, chromosomes, linear encodings of a problem's possible solution, are selected; operations such as crossover and mutation are applied; and they survive in higher probability if they are regarded as better ones.

GAs have been used to find good solutions to various NP-complete problems such as time-tabling problems [RCF94] and cable routing problems [KS94]; and have shown good performance in many applications. Actually some papers such as [CHMR91] and [Tam92a] suggested GA's superiority in FLPs. So, GAs may be a promising approach to FLPs.

Nevertheless, the GA's superiority on FLPs has not been clearly evaluated because the previous work only showed some good performance in a particular problem. That is, there has been no paper which compared GAs with other algorithms including stochastic approaches on various non-identical FLPs. Moreover, the effects of GA parameters such as crossover rates and mutation rates have not been sufficiently investigated for FLPs. Because GA performance usually depends on the GA parameters as mentioned in [Gol89] and [Dav91], the effects of many types of GA parameters are worth investigating.

1.3 Contribution of My Thesis

Accordingly, as my research work, I first established fifteen standard non-identical FLPs. Then, using the standard FLPs;

- I compared performance of GAs with other algorithms as well as with each other, and
- I investigated the effects of some GA parameters on FLPs.

Because the FLPs were picked up from previous papers where algorithms other than GAs were used, I believe the FLPs supply a fair basis for comparison.

To solve non-identical FLPs, the layout representation is generally important, since it influences the range of layouts which can be expressed. From many prospective representation methods, I chose the Slicing Tree Structure (STS) [Ott82] because STSs can represent wider range of layouts than other methods such as the cell assignment method in [BAV64] and the flexible bay structure method in [ST93]. Also, because STSs can be expressed by Polish expressions, STSs may be suitable for computation.

Among six types of GAs I implemented (Cea, Tam, DK, Tam2, DK2 and Kad algorithms), there are two duplicates of previous work ([CHMR91] and [Tam92a]), which used different chromosome representations from each other. While one of the two conventional GAs (the Cea algorithm) directly used Polish expression for the chromosome representation, the other (the Tam algorithm) used reduced Polish expression, which only contains the operators of the corresponding ordinary Polish expression. In order to use reduced Polish expression, the Tam algorithm limited the search space where the solutions were looked for by a clustering method. As regards the four other GAs (DK, Tam2, DK2, Kad), they can be regarded as modified versions of the Tam algorithm. That is, they used another clustering method (DK, DK2, Kad) and/or another chromosome representation to remove the search space limitation (Tam2, DK2, Kad).

By comparing the performance of GAs with other algorithms including simulated annealing and quasi-Newton methods, I confirmed that the performance of GAs was generally better than that of other algorithms. In particular, this superiority was significant in Tam2, DK2 and Kad algorithms using Genitor reproduction [Whi89] with producing twin children.

From the comparison of the performance of GAs with each other, I found that limiting initial search space by some reasonable clustering methods clearly improved the GA performance in FLPs consisting of many facilities. However, I also found that limiting search space during the search like the Tam and DK algorithms often caused premature convergence leading to poor solutions.

Regarding the investigation of GA parameters, I observed results in FLPs consistent with other GA studies in fields different from FLPs. For instance, it was confirmed that GAs with large population size got better solutions despite slow convergence speed, and that Genitor reproduction, especially when crossover produced two complementary children showed better performance than generation-based reproduction in general.

My thesis consists of seven chapters including this chapter. In Chapter 2, FLPs and GAs are reviewed and my research interests are mentioned. In Chapter 3, non-identical FLPs are surveyed and fifteen standard problems are specified. The implementation details of STSs and GAs are described in Chapters 4 and 5, respectively. Chapter 6 shows experiments and results. Finally in Chapter 7, the conclusions of my work and some suggestions for future work are given.

Chapter 2

A Review of FLPs and GAs

2.1 A Review of Facility Layout Problems

2.1.1 What is the Layout Problem?

Many sorts of layout problems can be mentioned. For example, bin-packing problems such as [Smi85] and [Fal94] try to maximise the number of packets in a storage; VLSI chip layout problems like [LGW92] and [SR91] aim to minimise the area occupied by chips. In these problems, there are usually some domain specific constraints to be considered. Further examples include the clampability, stability, etc. of stacked loads in pallet loading problems (e.g. [CD85], [SDC80]); each chip's input and output connecting points in VLSI layout problems; and the trim loss in cutting rectangular materials from a large sheet in cutting stock problems (e.g. [Agr93], [Rei93], [YZH91]).

In particular, facility layout problems (FLPs) may be one of the largest fields. This is because there is usually a problem to minimise total traffic cost between facilities in a building. For example, FLP can be applied to manufacturing machines in a factory [Sou93], to people traffic in a office [Ste87], and to backboard wiring on an electrical board [Ste61]. In FLPs, under given facility conditions including facility size and traffic between each pair of them, the problem is to minimise the traffic cost.

2.1.2 FLP formulation and NP completeness

Though some other models have been suggested later as shown in [KH87], the quadratic assignment problem described in [KB57] may be regarded as the basic representation of FLPs. In the model, placing M facilities into M locations was represented as a problem to find the layout minimising the following function value F .

$$F = \sum_{i=1}^{M}\sum_{j=1}^{M}(traffic)_{ij} \times (distance)_{f(i)f(j)} \tag{2.1}$$

$$\begin{aligned} \text{where } (traffic)_{ij} &= \text{the traffic between facilities } i \text{ and } j \\ (distance)_{kl} &= \text{the distance between locations of } k \text{ and } l \\ f(i) &= \text{the location of facility } i. \end{aligned}$$

As for distance measurement method, most FLPs use one of two methods below. The first one is called *rectangular* or *Manhattan* method, which is the sum of the vertical and horizontal distances between the locations. For instance, if facility No.1 is at (3.0, 8.0) and if facility No.2 is at (7.0, 5.0), then the rectangular distance is 7.0 ($because \mid 3.0 - 7.0 \mid + \mid 8.0 - 5.0 \mid = 4.0 + 3.0 = 7.0$). This method may be useful for building layouts where all the rooms are rectangular and corridors are situated along the walls of the rooms. The second one is called *straight* or *Euclidean* method, which calculates the geometric distance. In the above example, the straight distance is 5.0 ($because\sqrt{(3.0 - 7.0)^2 + (8.0 - 5.0)^2} = 5.0$). This method may be suitable for wiring problems, etc. In both of the above methods, the distance is usually measured from the centre of gravity of a location.

Because there are $M!$ ways of putting M facilities into M locations, the above function can take $M!$ different values at most. Accordingly, in order to get the best layout (i.e. the least F value), all the $M!$ patterns should be estimated. Nevertheless, since $M!$ grows extremely large if M goes large, it is impossible to search for all patterns in polynomial time. That is, the FLP consisting of many facilities is a NP-complete problem [SG76].

Table 2.1. A classification of FLP solution method

optimal method	looking for the best layout (How to prune the redundant alternative is important.) [Lan63], [GP66]
suboptimal method	looking for reasonably good layouts
– constructive approach	putting one facility to another (Usually facilities having heavier load are considered prior to lighter ones.) [EGH70], [Neg74]
– improving approach	from a random layout, exchanging some selected facilities (Selection strategy is important.) [BAV64], [HC66], [NVR68]
– hybrid approach	a combination of constructive and improving approach [BS78], [BK83], [SV85]
– graphical approach	analytical approach (a sort of constructive approach) [Fou83], [Leu92]

2.1.3 Optimal and Suboptimal Algorithms

To tackle the FLP, many methods have been suggested. As shown in Table 2.1, the methods can be classified into two groups: optimal methods and suboptimal methods. Optimal methods search for the best answer by some heuristics like branch and bound in [Lan63] or [GP66], whereas suboptimal methods look for reasonably good solutions by various strategies. However, owing to the NP-completeness, the approaches for optimal solution may be impractical especially when the number of facilities is big. [HK72]

According to [KH87], the suboptimal methods can be categorised into four approaches as shown in Table 2.1. Some algorithms like MAT in [EGH70] and LPA in [Neg74] are called constructive or additive approaches, since they make physical layouts by adding one facility to another. In these approaches, deciding the order of putting facilities is generally important. For example, LPA first puts a pair of facilities, which have the highest traffic; then, put another facility, which have the highest traffic with the facilities already located, as close as possible to them; and so on. MAT initially sorts each pair of facilities in order of traffic quantity; then, put each pair into appropriate vacant spaces in the order.

Improving approaches such as CRAFT in [BAV64] and HC63-66 in [HC66] repeatedly exchange the places of some facilities in order to get better layouts. This repetition starts from an initial layout generated at random until either a reasonably good solution is obtained or a certain allowed time passes. In these approaches, the selection strategy for exchanging facilities is critical. For instance, while CRAFT has no limitation for choosing facilities to be exchanged, HC63-66 restricted exchanging facilities to reduce calculation time. Also, some algorithms like Biased Sampling Method in [NVR68] select facilities stochastically to save time.

Hybrid approaches can be said to have the merits of both approaches above. That is, starting from a layout created by a constructive algorithm, the layout is modified by some improving algorithms. For example, [BS78] creates an initial layout by branch and bound method under time limits; then, improves it by exchanging two or three facilities at one time.

In contrast, graphical approaches such as [Fou83] and [Leu92] have a rather different point of view. These analytical approaches put importance on the adjacency of facilities, since the definition of $(distance)_{ij}$ of Formula (2.1) is generally changed as follows.

$$(distance)_{ij} = \begin{cases} 0 & \text{if facilities } i \text{ and } j \text{ are adjacent} \\ 1 & \text{otherwise} \end{cases}$$

In these approaches, each facility and each adjacency of two facilities are represented by a node and an arc, respectively. Therefore, a feasible layout must be a planar graph, which can be drawn on a plane without any intersections of lines. Thus, as [KH87] suggested, these approaches may be regarded as sorts of additive approaches because they construct a larger graph by adding a new node to a planar graph so that the new graph can be still planar and that its traffic cost can be as small as possible. However, according to [Fou83], these approaches may be able to solve FLPs with at most fifteen facilities in reasonable computation time.

6	10	2	3
5	1	8	11
9	12	4	7

Figure 2.1. Cell assignment representation for an identical FLP

2.1.4 Problems Variation

Other than the algorithms classification above, FLPs research can be categorised from many aspects. Here, some of them will be introduced.

Identical/Non-identical Facilities and Layout Representations

Although many FLPs consist of facilities of the same shape (*identical* facilities), some FLPs include the facilities of different shapes (*non-identical* facilities).

In the former case, the *cell assignment* method shown in Figure 2.1, where a certain area is first divided into identical cells and each facility is assigned to one of the cells, may be reasonable to represent physical layouts. But, in the latter case, this method is not always suitable. For instance, [BAV64]'s method enables one facility to be assigned to more than one cell as shown in Figure 2.2. However, because this method often creates facility regions having strange shapes, it may be unsuitable for practical use. Besides, [SLMK92] considered the possibility of using conventional cell assignment representation, even if the facilities contain non-identical ones; nevertheless, this method usually generates useless gaps between facilities. Hence, many other layout representation methods have been reported as alternative approaches for non-identical FLPs. Here, *multi-row representation* [SLMK92], *flexible bay structure* [ST93] and *slicing tree structure* (STS) [Ott82] can be mentioned.

In the multi-row representation, each facility is first assigned to a certain grid position; then, the facilities are pushed toward a certain corner as shown in Figure 2.3. In the flexible bay structure, a particular room, to which all the

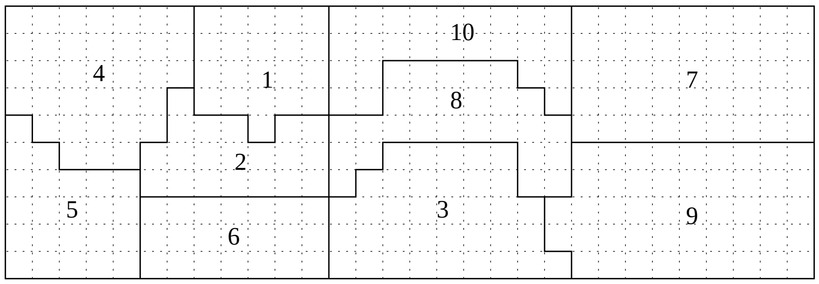


Figure 2.2. Cell assignment representation for a non-identical FLP

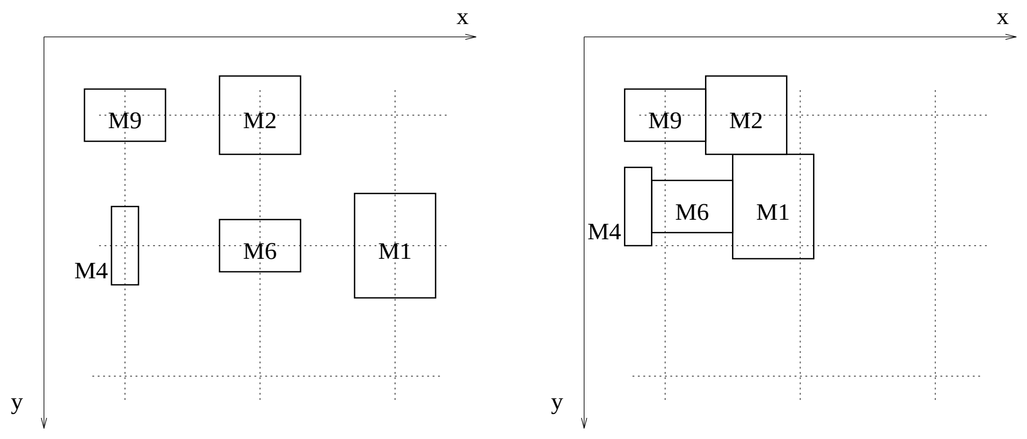
facilities are assigned, is first divided into some bays; then, each bay is separated into cells as shown in Figure 2.4. Since the lines separating cells and/or bays can flexibly move inside the room, each cell is able to have the required area.

In the STS, a layout is represented by a binary tree, where each terminal node corresponds to a facility and each non-terminal node indicates the relative position of facilities. For instance, the layout shown in Figure 2.5(a) can be represented by the tree shown in Figure 2.5(b). Here, the numbers in the terminal nodes express the facility's index and the letters in the non-terminal nodes express the relation of each substructure, where the relation is one of the following four:

`U`	=	*"The substructure given by the second argument is just above the substructure given by the first argument."*
`B`	=	*"The substructure given by the second argument is just beneath the substructure given by the first argument."*
`L`	=	*"The substructure given by the second argument is just left of the substructure given by the first argument."*
`R`	=	*"The substructure given by the second argument is just right of the substructure given by the first argument."*

Also, an STS can be expressed by a Polish expression, where terminal nodes and non-terminal nodes are considered as operands and operators, respectively. So, the layout in Figure 2.5(a) can be represented by the Polish expression `43R62U5R1LB`.

To decode a Polish expression, we may assume a stack machine which works based on the procedure shown in Table 2.2. For example, a Polish expression



Step1. Assignment of the machines to the sites

Step 2. Bringing the machines together

Figure 2.3. Multi-row representation

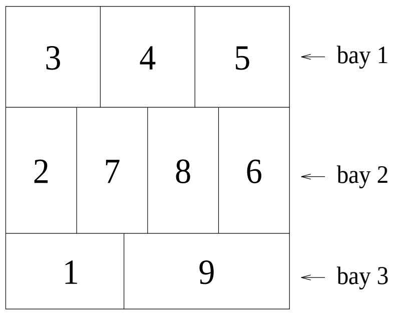


Figure 2.4. Flexible bay structure

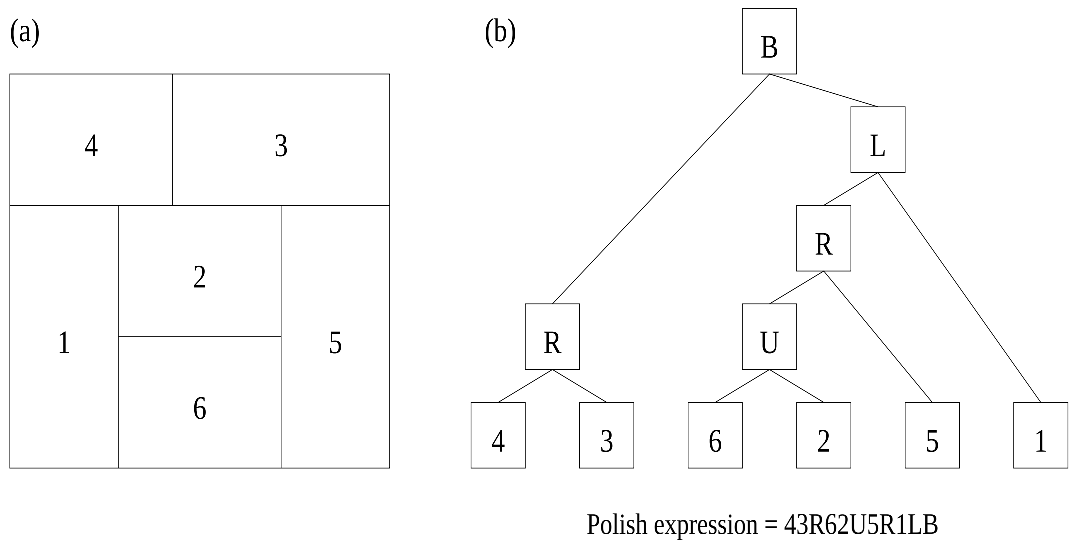


Figure 2.5. Slicing tree structure (STS)

`43R12UL` will be decoded as shown in Figure 2.6.

Layout Constraints Regarding the shape of facilities, some layout constraints have been taken into account. Here, aspect ratio, dead-space ratio, and prespecified area will be introduced.

Aspect ratio of the assigned area, which is the ratio of its vertical length to its horizontal length, may be important for FLPs. This is because if a facility requiring $20m^2$ is assigned to the area of $0.1m \times 200m$, the facility area may be useless. Therefore, many FLPs set a certain limitation for each facility's aspect ratio (e.g. [CHMR91], [Tam92a], [Tam92b]). Of course, the more rigid the limitations are, the more difficult the problem will be [KJK91].

Dead-space ratio for a facility was suggested by [Tam92a] and [Tam92b]. It is calculated by the following formula.

$$(\textit{dead-space ratio}) = 1 - \frac{AS}{RS}$$

where

AS = assigned space

RS = minimum rectangular space including the assigned space

Table 2.2. A decoding procedure for a Polish expression

step 1:	Read each symbol in the Polish expression from left to right.
step 2:	If an operand appears, push it in the stack.
step 3:	If an operator appears, pop two items from the stack, create a binary tree by putting the operator as its root and by using two items as its terminals, and push the tree in the stack.
step 4:	Until all the symbols in the Polish expression are read, repeat steps 2 and 3.
step 5:	Pop the final result from the stack.

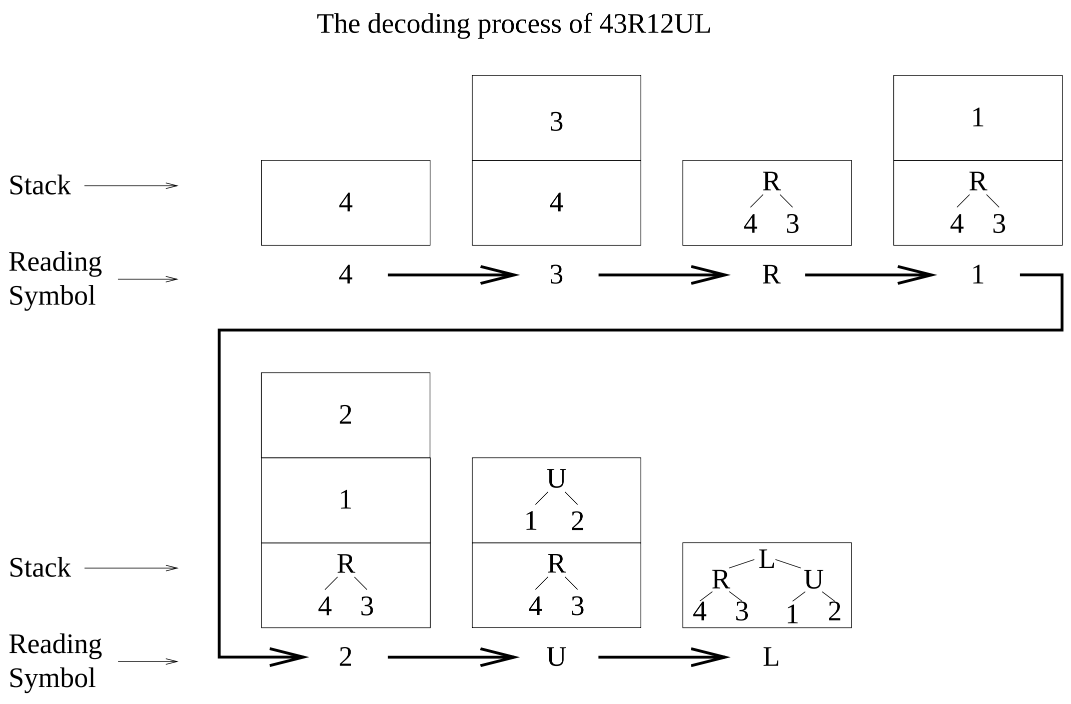


Figure 2.6. An example of decoding process of a Polish expression

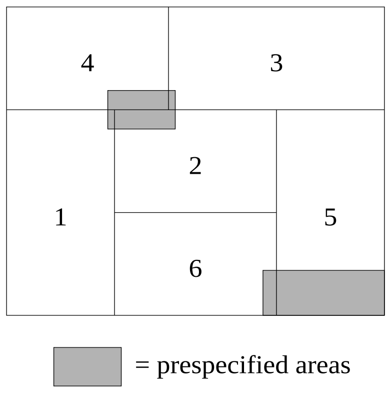


Figure 2.7. A layout in an FLP with prespecified areas

That is, if the assigned area is rectangular, the value will be 0; and if the area is not rectangular (e.g. L-shape), the ratio will be between 0 and 1. Therefore, if a particular facility can be fitted in a non-rectangular area, the possible range of dead-space ratio for the facility will be wide. So, this limitation may be also useful to specify the constraints of each facility.

In contrast, *Prespecified areas* mentioned by [Tam92a] and [Tam92b] concern the constraints of the room, where facilities will be put, rather than those of the facilities. Because the room may have some reserved space for utilities, pillars and so on, such space should be excluded from locating facilities. For example, Figure 2.7 shows a layout corresponding to the STS shown in Figure 2.5(b) in an FLP with prespecified areas.

Layout Dimensions Although most FLPs address two dimensional layout problems, some of them introduced other dimensions as well. SPACECRAFT in [Joh82] and [LM81] showed three dimensional layout problems for allocating departments into a multi-storey building. While SPACECRAFT took an improving approach by adapting CRAFT in [BAV64] to three dimensional FLPs, [LM81] put importance on the constructive approach by using the probabilistic idea of [GW70] which takes account both of immediate cost and of future cost. The immediate cost means the actual traffic cost caused by putting a particular facility into a location, and the future cost concerns probabilistic cost due to the restrictions caused by putting the facility to the place.

In contrast, [MV91] showed an example of facility layout problems where facilities should follow both sides of one road; and [KB87] mentioned a case of VLSI chip layout problems where facilities should follow some rows.

Thus, other dimensional problems than two dimensions may be important for practical use and it may need some different ideas to solve the problems. For example, [Joh82] mentioned that the traffic cost calculation in multi-storey building problems usually takes much time. This is because alternative routes are frequently found concerning the location of lifts, etc., when some facilities are swapped over in the stage of layout improvement. However, the basic ideas may be common for those problems; accordingly, I think the investigation of two dimensional FLPs may be relevant to other dimensional FLPs.

Other Variations [Ros79] and [DS82] introduced FLPs with multi-goals. Though ordinary FLP only takes traffic cost into account, they considered FLPs in which the adjacency of particular pairs of facilities was of extra importance. Suggesting the following formula, [DS82] mentioned that suitable layouts should be decided with the consideration of the balance of factors (e.g. W_1 and W_2).

$$\begin{aligned}
\text{Minimise: } C &= W_1 \times F - W_2 \times R \\
\text{subject to} & \\
F &= \text{conventional traffic cost (See Formula (2.1))} \\
R &= \sum_{i=1}^{M}\sum_{j=1}^{M}(closeness)_{ij} \times (adjacency)_{ij} \\
W_1, W_2 &= \text{weights for } F \text{ and } R \text{ where } W_1 + W_2 = 1 \\
M &= \text{the number of facilities} \\
(closeness)_{ij} &= \text{the degree of importance of the adjacency} \\
& \quad\ \text{of facilities } i \text{ and } j. \\
(adjacency)_{ij} &= \begin{cases} 1 & \text{when facilities } i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

For example, C can be made smaller by making R larger and R can be made large by ensuring that facilities which ought to be close are adjacent.

[Ros86], [MV91], [Urb92], etc. regarded FLPs as a dynamic problem rather than a static problem. For example, [Ros86] gave a formulation of dynamic FLPs as follows because traffic may change over time.

$$\begin{aligned}
\text{Minimise: } L_T &= \sum_{t=1}^{T}(C_{t-1,t} + Z_t) \\
\text{subject to} & \\
L_t &= \text{total costs for all periods up to } t \\
C_{s,t} &= \text{rearrangement costs for layout used in period } s \\
& \text{ to that used in period } t \\
Z_t &= \text{conventional traffic costs for layout used in period } t
\end{aligned}$$

Some research put importance on the user interface. [RR91] used fuzzy logic to represent the traffic cost matrix table. This may be helpful to establish the table, when interviews are conducted with domain experts who may only have uncertain knowledge of the traffic quantity between each pair of facilities. [BMMK92] mentioned a system which can propose alternative layouts based on the feedback of users.

In conclusion, these variations formulated new types of FLPs and/or addressed a new aspect of FLPs. So, they may be useful when particular practical problems are solved.

2.1.5 Recent Research

So far, some survey researches for FLPs have been reported. [NVR68] suggested eight standard problems, which have been used by many researchers for bench mark tests, and compared four suboptimal algorithms. Similarly, [KH87] and [YP93] used the eight standard problems and compared twelve and ten algorithms, respectively. However, they might be still insufficient because of the following two points.

First, most of the algorithms compared in the papers relied on deterministic approaches. Because deterministic approaches such as hill-climbing methods may only reach one of many local minima, the final solution might not be sufficiently good. As some recent researches in Table 2.3 suggested, stochastic approaches such as simulated annealing (SA) [Egl90] and genetic algorithms (GAs) could be used to obtain better solutions. Because these stochastic approaches search for wider alternatives by considering even worse solutions, they may reach better solutions efficiently. Unfortunately, most of the researches in Table 2.3 only stated that their own methods showed better solutions in a particular problem. Therefore, it is still unclear which algorithm in Table 2.2 shows the best performance. That is, a survey could be done including the new algorithms.

Second, the above survey papers did not use non-identical FLPs for their benchmarks. Because non-identical FLPs may often appear in practical situations, the comparison of the algorithms for non-identical FLPs is probably valuable. On the other hand, the researches in Table 2.3 solved non-identical FLPs using some unique layout representation techniques. For instance, [CHMR91] etc. used the slicing tree structures (STSs) (Figure 2.5); [Sou93] etc. used the multi-row representations (Figure 2.3); and [ST93] used the flexible bay structure (Figure 2.4). [VCCV91] and [TL91] suggested a representation method which may be called circle-to-rectangle. For example, in the method of [TL91], each facility's shape is first considered as a circle whose area is equal to its required area; then, the circles are assigned to suitable places with a quasi-Newton procedure by assuming an attractive force proportional to the traffic between facilities and a repulsive force preventing facilities to overlap; then, the circles are converted to rectangles so that they can not intersect each other, that they can retain their areas the same, and that they can keep their position as much as possible.

In conclusion, in spite of the fact that the new algorithms in Table 2.3 are hopeful due to their stochastic approaches, they have not yet been carefully compared. Hence, it will be useful if these new algorithms are compared on the same problems including non-identical facilities and if their performance are compared with other sorts of approaches.

Table 2.3. Algorithms for FLPs with non-identical shapes

from	category	layout representation
[CHMR91]	genetic algorithm	slicing tree structure
[Tam92a]	genetic algorithm	slicing tree structure
[ST93]	genetic algorithm	flexible bay structure
[WL86]	simulated annealing	slicing tree structure
[KJK91]	simulated annealing	slicing tree structure
[Tam92b]	simulated annealing	slicing tree structure
[Sou93]	simulated annealing	multi-rows representation
[KB87]	simulated evolution	multi-rows representation
[VCCV91]	quasi-Newton	circle-to-rectangle
[TL91]	quasi-Newton	circle-to-rectangle

2.2 A Review of Genetic Algorithms

2.2.1 What is a Genetic Algorithm?

Genetic algorithms (GAs) are a problem solving technique hinted at by living creatures' evolution [Whi93]. In GAs, chromosomes, linear encodings of a problem's possible solution, are selected from a population; operations such as crossovers and mutations are applied; and they survive in higher probability if they are regarded as better ones. That is, a GA's mechanism is similar to nature's one in which superior individuals can produce more descendants in the future. A typical flow diagram of a GA which is called *generation-based reproduction* method is shown in Table 2.4.

In order to use GAs for solving a problem, important points are: representation of the chromosomes; design of crossover and mutation operators; and fitness functions [Whi93] [Gol89].

For instance, suppose that we are trying to use GA for finding minimum z value where $z = x_1^2 + x_2^2 + x_3^2 + ... + x_k^2$ and $x_i (1 \leq i \leq k)$ is a real number. At that time, we may define: the representation of chromosome is $\mathbf{x_1}$-$\mathbf{x_2}$-$\mathbf{x_3}$-$\cdots$-$\mathbf{x_k}$; the crossover of two parents Pa and Pb, represented by $\mathbf{a_1}$-$\mathbf{a_2}$-$\mathbf{a_3}$-$\cdots$-$\mathbf{a_k}$ and $\mathbf{b_1}$-$\mathbf{b_2}$-$\mathbf{b_3}$-$\cdots$-$\mathbf{b_k}$ respectively, produce a child C, represented by $\mathbf{c_1}$-$\mathbf{c_2}$-$\mathbf{c_3}$-$\cdots$-$\mathbf{c_k}$ where c_i is either a_i or b_i; the mutation changes the child to $\mathbf{d_1}$-$\mathbf{d_2}$-$\mathbf{d_3}$-$\cdots$-$\mathbf{d_k}$

Table 2.4. A flow diagram of GA (generation-based reproduction)

step 1:	set up initial chromosomes at random
step 2:	select two chromosomes
step 3:	crossover them to produce a child
step 4:	mutate the child
step 5:	repeat steps 2 to 4 until the number of children becomes equal to that of parent's generation.
step 6:	replace the parent's generation with the children and regard it as new generation.
step 7:	repeat steps 2 to 6 until either all the chromosomes are same (i.e. converged) or a best solution already known has appeared, or enough time has passed.

where d_i is $c_i + e_i$ and e_i is a random number. Also, we can use $1/(z+1)$ as the fitness function because smaller z makes the fitness function's value bigger and because the function's value is still valid if z becomes 0.

2.2.2 GA parameters and performance

In GAs, there are many kinds of parameters, which influence the GA's behaviour. Here, some important GA parameters and their influences will be briefly reviewed.

Crossover and Mutation Crossover usually takes two parents and can produce one, two or more children. In actual GAs, the allele of either parent is simply copied into the corresponding place of the child's chromosome. There are some variations e.g. one-point crossover, two-point crossover and uniform crossover. *One-point crossover* first specifies a split point on a chromosome at random; then copies the alleles between the head and the splitting point of one parent and those between the splitting point and the tail of the other parent. *Two-point crossover* initially chooses two splitting points; then duplicates the alleles between the splitting points of one parent, and the other alleles from the other parent. *Uniform crossover* randomly picks each allele from either of the two parents. For example, if two parents `1-2-3-4` and `5-6-7-8` are selected, a child `1-2-7-8` may be created by one-point crossover, `5-2-3-8` may be produced by two-point crossover, and uniform crossover may generate `1-6-3-8`.

In contrast, mutation may happen to one selected allele; or probabilistically to every allele. For instance, a chromosome `1-2-3-4` may be changed to `1-2-5-4`.

At the steps 3 and 4 of Table 2.4, *crossover rate* and *mutation rate* are applied. For example, if crossover rate is 0.6 and mutation rate is 0.01, crossover will happen with 60% probability and mutation will occur on each allele with 1% probability. So, if the chromosome consists of L alleles, a chromosome's changing probability by the mutation is $1-(1-M)^L \approx LM$ where M is the mutation rate and M can be assumed to be much smaller than $1/L$. [Gol89]

Population Size The number of chromosomes is often called *population size*, and it may also influence the GA. As mentioned in [Gol89], a GA of large population size may have better solutions ultimately because of large number of chromosomes may include good schemata in some chromosome. On the other hand, GAs with smaller population can change rapidly; therefore, it may show better performance in the early stages rather than those with a larger population.

In parallel GAs, where there are separately evolving populations which occasionally exchange a chromosome, *the number of populations* may be an influential factor. [CHMR91]

Selection Methods Among various kinds of selection methods, rank and two types of tournament selection methods will be introduced here.

Rank method [Bak85] first sorts all the chromosomes in order of fitness values; then, the probability of selecting a particular chromosome is proportional to the inverse for the order rather than the fitness itself. For instance, if there are four chromosomes whose fitness values are 1, 5, 7, 3; then the rank ordering is 4th, 2nd, 1st, 3rd and the probabilities of selection are $\frac{1}{10}, \frac{3}{10}, \frac{4}{10}, \frac{2}{10}$.

In contrast, *tournament selection* [Bri81] is as follows. First, a particular number S is decided as the size of tournament. Second, S chromosomes are uniformly chosen from all the chromosomes. Finally, the best one among the S chromosomes is selected as a parent. Of course, two parents are required in normal GAs; therefore, the above process is usually done $2N$ times, where N is the population size. In the tournament selection, the same chromosome may be chosen more than once [Whi93].

However, the tournament selection with large S causes a strong pressure to choose very fit chromosomes. And, this leads premature convergence of chromosomes which usually produce only poor solutions. To explain it, let us consider the probabilities of being chosen as a parent for the five chromosomes: the best one in the generation; the 75th percentile; the median; the 25th percentile; and the worst one. In tournament selection, each candidate for parents has to win against $(S-1)$ competitors in a group to become a parent. Because the probabilities of meeting a weaker chromosome for the five chromosomes are 100%, 75%, 50%, 25% and 0%; the probabilities of becoming a parent by winning against $(S-1)$ competitors for the five are 1, $(0.75)^{S-1}, (0.5)^{S-1}, (0.25)^{S-1}$ and 0. Thus, if $S = 2$, they are 1, 0.75, 0.5, 0.25, and 0; and if $S = 5$, they are 1, 0.32, 0.06, 0.004 and 0. Hence, we can see that very fit chromosomes will be frequently chosen as parents in large S.

Modified tournament selection method may be useful in some cases to cope with this defect [RH95]. In this method, a chromosome is first chosen as the first candidate at random; secondly, the chromosome is compared with at most $(S-1)$ chromosomes randomly chosen; if a better chromosome than the first candidate is found from the $(S-1)$ chromosomes, then the better one is selected as a parent immediately; however, if all the $(S-1)$ chromosomes are worse than the first candidate, then the first one is selected as the parent. Therefore, other candidates than the first one can become the parent only by beating the first candidate. Hence, the strong pressure to choose very fit chromosome observed in tournament selection should become weaker in this modified tournament selection.

Reproduction Methods There are some variations of reproduction methods. That is, the flow diagram of Table 2.4 may be changed.

For instance, in *Genitor* method [Whi89], which is sometimes known as steady state reproduction [Dav91], steps 5 and 6 of Table 2.4 are replaced with step $5'$ below.

step $5'$: replace the worst chromosome in parent's generation with the child if the child is better than the worst one.

Genitor method can be considered as one of the $(\mu + \lambda)$ evolution strategy [BHS91], in which λ offspring are produced from μ parents and the best μ chromosomes of $(\mu + \lambda)$ are retained. Because the best chromosomes are always retained in this strategy, the population may converge gradually without drastic drifts. On the other hand, the generation-based GA is regarded as one of the (μ, λ) evolution strategy, in which λ offspring are produced from μ parents and the best μ chromosomes of λ are retained. Because the best chromosomes may be lost in this strategy, the population may dramatically drift in search space for solutions. Therefore, all the chromosomes in Genitor may converge (become the same) quicker than those in the generation-based GA [Dav91]; but Genitor may produce only poor solutions due to premature convergence [Whi93].

It is also possible for the reproduction step to produce more children rather than one. For example, twin children which have complementary alleles of parents may be produced at the crossover stage. And in some GAs (e.g. [YP93] and GIGA in [Cul92]), reproduction produces many children and only the best few of those are then kept.

On-line, Off-line and Best Individual Performance According to [Bak85], there are three types of criteria to evaluate the GA performance. They are: *the on-line performance*, the average of all results that have appeared; *the off-line performance*, the average of best results of each generation; and *the best individual performance*, the best result that has appeared. In other words, they can be shown in the following formulae.

$$\begin{aligned}(on\_line)_T &= \frac{1}{T \times N} \sum_{i=1}^{N} \sum_{t=1}^{T} (fitness)_{t,i} \\ (off\_line)_T &= \frac{1}{T} \sum_{t=1}^{T} (best\_individual)_t\end{aligned}$$

where

$$\begin{aligned}
(on\_line)_t &= \text{on-line performance at time } t \\
(off\_line)_t &= \text{off-line performance at time } t \\
(best\_individual)_t &= \text{best individual performance at time } t \\
&\quad \text{(i.e. the best score up to time } t\text{)} \\
(fitness)_{t,i} &= \text{the fitness value of the } i\text{-th chromosome at time } t \\
N &= \text{population size}
\end{aligned}$$

While the off-line and the best individual performances only take account of the best chromosome in each generation, the on-line performance reflects the performance of chromosomes other than the best one as well. Therefore, the GAs showing good on-line performance may not produce remarkable chromosomes.

Between the off-line and the best individual performances, the off-line one can take the convergence speed into account unlike the best individual performance. For example, if two algorithms A and B show the best individual performance as shown in Fig 2.8, the off-line performance of algorithm A is better than that of B at time $T1$, though the best individual performance of both algorithms are same. But, at time $T2$, the off-line performance of algorithm A is still better than that of B, although its best individual performance is worse. That is, off-line performance is generally influenced by the past records.

However, because the quality of the best solution may be important in practical applications, the best individual performance may be more useful than the other performance measures. Hence, the best individual performance is used in this thesis.

Genetic Programming As a similar paradigm of GAs, Genetic Programming (GP) can be mentioned. As [Koz92] introduced, GP mainly concerns producing the fittest computer programs. However, GP can be regarded as an extension of GAs because the solutions appeared in GP do not have to be fixed-length unlike GAs. So, various technique for GP may be also useful for GAs.

Moreover, GP usually uses a tree structure consisting of operators and operands, which represents programs, as the representation of solutions. Therefore, facility layout representations such as Slicing Tree Structure might be enhanced by using

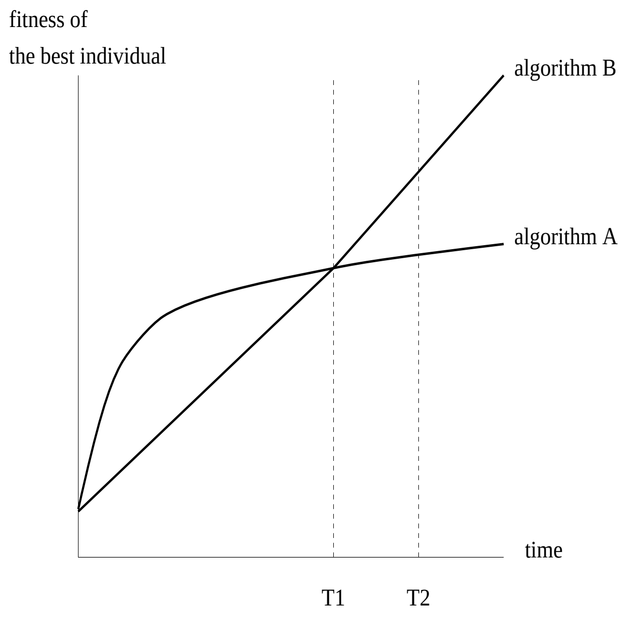


Figure 2.8. A sample of two algorithms' performance

GP related ideas.

2.2.3 GA Applications

As [GG89] mentioned, GAs may be effective approaches for NP-complete problems as ones of stochastic approaches. For example, [FRC93] and [RCF94] introduced GA's efficiency on job-shop scheduling problems and time tabling problems, respectively. Similarly, various layout problems and FLPs are solved by GAs. Here, I will review some of them.

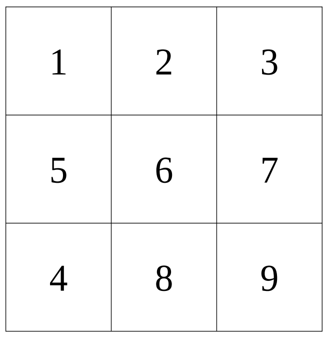
Layout Problems with GAs Regarding layout-related research, [KS94] tackled cable routing problems with a GA. In the GA, a chromosome consists of the index of each cable's routing alternatives. For example, if there are three cables to be routed, a chromosome will have three alleles. And, if a chromosome is `2-3-2`; the first cable will be routed by the second possible way for the cable, the second cable will be routed by the third possible way for the cable, and the third cable will be routed by the second possible way for the cable. Although [KS94] did not report any details how each cable's alternative ways are produced, it suggested the GA worked well.

Whereas [KS94] was able to use traditional crossovers and mutations, [Smi85] had to use a modified crossover to cope with his chromosome encoding in a bin-packing problem. In his research a chromosome represents a list of packing order of objects. For instance, if a chromosome is `4-1-3-2`; then it represents that object No.4 will be first packed, object No.1 will be second packed, and so on. Therefore, ordinary crossovers may produce nonsense children. For example,if `4-1-3-2` and `1-2-3-4` are one-point crossed over, and if a splitting point is set between the second and third genes; then, it may produce twin children `4-1-3-4` and `1-2-3-2`, which do not represent solutions of this problem. In order to tackle this problem, he used a modified crossover which keeps the genes before splitting point of the first parent and applies the order in the second parent for the rest of objects. So, in the above example, `4-1-2-3` and `1-2-4-3` may be created instead.

Furthermore, [Fal94] took account of the redundancy of the representation of chromosomes. For example, in the encoding of [Smi85], `1-2-3-4` and `4-3-2-1` may be different solutions. Nevertheless, if objects 1 and 2 fill a bin and if objects 3 and 4 fill another bin, these two chromosomes virtually represent the same solution. Because higher redundancy makes the GA's search space larger and the GA's power weaker, he suggested another encoding method to reduce the redundancy. Although it might be highly dependent on the problem's domain, we may be able to see the fact that chromosome's representation and design of crossovers and mutations will much influence GA performance.

FLPs with GAs As regards FLPs, various kinds of chromosome representations and crossovers and mutations have been also suggested.

[CP87] introduced an original crossover method for the cell assignment representation. For example, in a 3×3 FLP, a chromosome `1-2-3-5-6-7-4-8-9` may represent a layout shown in Figure 2.9. Since the conventional crossover tends to favour shorter schemata more [Whi93], the relation of facilities No.5 and No.6 may be kept in higher probability than that of No.2 and No.6, in this example. However, in FLPs, other dimensional adjacency (i.e. vertical adjacency in this example) may be as important as the encoding dimension's adjacency (i.e. horizontal adjacency in this example). [CP87] introduced a special crossover which



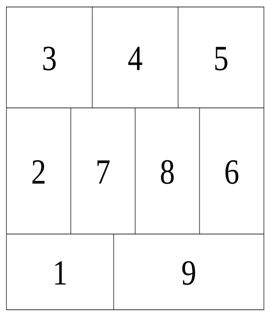
chromosome = 1-2-3-5-6-7-4-8-9

Figure 2.9. An example of cell assignment representation

can take into account such two dimensional adjacencies.

[ST93] used flexible bay structure representation to tackle non-identical FLPs. In the representation, a physical layout is represented by two chromosomes. The first chromosome specifies the order of putting facilities into cells, and the second one specifies how many cells are included in each bay (row). For instance, if the first chromosome is `3-4-5-2-7-8-6-1-9` and if the second one is `3-4-2`, the layout will be as shown in Figure 2.10. [ST93] used the crossover as follows: The child's first chromosome is produced from the first chromosomes of parents by the same method of [Smi85]; and the child's second chromosome is copied from either parent's second one. As for the mutation, one of the following three types is done. If the first type (MU1) is applied, a bay chosen at random is divided into two bays. If the second type (MU2) is applied, two sequential bays chosen at random are merged into one bay. If the third type (MU3) is applied, a part of genes will be reversed. That is, MU1 and MU2 affect the second chromosome, whereas MU3 affects the first chromosome. [ST93] set the probability ratio of MU1, MU2 and MU3 occurring to be 1:1:2.

While the flexible bay structure requires each cell to lie in rows (bays), the slicing tree structure (STS) can generate more various physical layouts as shown in Figure 2.5. However, if Polish expression, which corresponds to a tree structure and to a layout, is directly used as a chromosome's representation, ordinary crossovers and mutations can not be applied, because a combination of operators and operands in random order may not be a valid Polish expression. For example,



the first chromosome = 3-4-5-2-7-8-6-1-9
the second chromosome = 3-4-2

Figure 2.10. An example of flexible bay structure

if 12U3B and 312RL are one-point crossed over and if the splitting point is chosen between third and fourth genes, the children will be 12URL and 3123B which are not valid Polish expressions and do not correspond to any layouts.

In order to use Polish expression as chromosome's representation, [CHMR91] suggested several types of special crossover and mutation methods. For example, one of crossover methods creates a child so that it can inherit the tree's structure from one parent and that it can inherit the operators in the Polish expression from the other parent. As for mutations, [CHMR91] used [WL86]'s methods which are used for solutions move in simulated annealing. They are: swapping adjacent operands; switching a sequence of adjacent operators; and swapping an operator and a neighbourhood operand.

On the other hand, in order to use conventional crossovers and mutations, [Tam92a] suggested a method where the tree structure is fixed and a chromosome includes only operators of the Polish expression. For instance, a layout shown in Figure 2.5 is represented by a chromosome of RURLB because numbers 1 to 6 of 43R62U5R1LB are omitted. In other words, [Tam92a] limited the search space, while [CHMR91] did not.

In conclusion, there have been many methods for representation, crossovers and mutations; and this might suggest that better methods may appear in the future. However, so far, I think STS may be the most suitable way for layout representation because it can express various shapes and because the data structure of Polish expression, which can represent a layout, may match GAs. In addition, it may be interesting to investigate the hybrid techniques of GAs and STSs, since this sort of research may not have been done yet.

2.3 My Research Interests

In previous sections, I mentioned GA may be superior for NP-complete problems including FLPs. However, the comparison of algorithms for non-identical FLPs may be still insufficient because many papers only stated that some methods showed better solution in a particular problem. Therefore, I wish to propose some common non-identical FLPs as a standard for benchmark tests, and compare the performance of GAs against other methods using the standard problems.

In addition, the investigation of STS will be valuable for the following reasons. First, STS may be the most suitable way to represent a layout so far, because it can express more varied layouts than other representation methods. Second, as STS technique seems to be insufficiently investigated yet, some useful technique may be found. Third, GA and STS may match well due to the similar data structure, since they use a string of symbol for a chromosome or a Polish expression, respectively, to represent a solution.

Finally, I am also interested in testing GA parameters' effects on FLPs. As I reviewed, there are many kinds of GA parameters which may be influential on the performance. For example, [SCED89] reported that some special parameters combinations may be effective independently of problems. In the paper, they solved problems including Dejong's problems [Gol89], travelling salesperson problems, under various combinations of GA parameters as shown in Table 2.5. Focusing on on-line average performance, they reported some interesting results. For example, they found a strong interaction among population-size, crossover-rate and mutation-rate (pcm), while there was no relation found between problems and pcm. Thus, they suggested that a relation among pcm might be independent of problems. However, as already mentioned in the previous section, on-line performance may not be a serious measure for considering the GA parameters effects. Nevertheless, there might be some superior combinations of parameters in FLPs. Hence, I believe such GA parameters investigation may be valuable for FLPs.

In conclusion, my research interests can be summarised as follows.

Table 2.5. GA parameters investigated by [SCED89]

crossover rate	0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95
mutation rate	0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1
population size	10, 20, 30, 50, 100, 200
crossover points	one point, two points

- Investigating GA parameters to find out if there are some special combinations which are effective to FLPs independent of specific problems
- Comparing GA performance with other algorithms based on the standard problems
- Comparing different GAs performance with each other from the perspective of STS usage

Chapter 3

A Survey of Non-identical FLPs

3.1 A Survey of Recent Research

As mentioned in previous chapters, some past researchers have studied various non-identical facility layout problems (FLPs). The following Table 3.1 shows a list of some of these including information about the problems.

As shown in the table, two of six authors used Genetic Algorithms (GAs), another two relied on Simulated Annealing (SA), and the other two used quasi-Newton (QN) methods where the minimum traffic cost was sought under some constraints given by equations. Regarding representation of facilities, slicing tree structures (STSs) were used in the methods oriented for SA and GAs; whereas two QN approaches used circle-to-rectangle representations (CtoRs), where each facility is first assumed as a circle of required area and then transformed to rectangular shape. And, the number of facilities varied from five to thirty.

Although some research results were compared with other researches or other algorithms implemented by each author, there have been no common problems for all of them so far. Therefore, it seems to be difficult to say which algorithm is better. So, in order to compare these algorithms' performance with GAs, I decided to implement fifteen problems which are marked as $\star$ in Table 3.1. Because the same algorithms of [CHMR91] and [Tam92a] will be implemented as the part of my work (this will be described in the next chapter), a comparison between the GAs used in this thesis and earlier results will be possible.

Table 3.1. A list of problems concerning non-identical FLPs

	algo- rithm	repre- sentation	facility number	problem sources	compared with	problem index
[KJK91]	SA	STS	11	[KJK91]	-	⋆1
			11	[KJK91]	-	⋆2
			16	[CHMR91]	[CHMR91]	⋆3
			20	[CHMR91]	[CHMR91]	⋆4
[CHMR91]	GA	STS	16	[CHMR91]	SA	⋆3
			20	[CHMR91]	SA	⋆4
[TL91]	QN	CtoR	5	[TL91]	-	⋆5
			6	[TL91]	-	⋆6
			7	[TL91]	-	⋆7
			8	[TL91]	-	⋆8
			12	[TL91]	-	⋆9
			15	[TL91]	-	⋆10
			20	[TL91]	-	⋆11
			30	[TL91]	-	⋆12
[Tam92a]	GA	STS	12	[Tam92a],[Tam92b]	HC	
			15	[Tam92a],[Tam92b]	HC	
			20	[Tam92a],[Tam92b]	HC	⋆13
			30	[Tam92a],[Tam92b]	HC	⋆14
[Tam92b]	SA	STS	20	[Tam92a],[Tam92b]	HC	⋆13
			30	[Tam92a],[Tam92b]	HC	⋆14
[VCCV91]	QN	CtoR	10	[VCCV91]	-	⋆15

SA = simulated annealing
GA = genetic algorithm
QN = quasi-Newton method
HC = hill climbing method
STS = slicing tree structure
CtoR = circle-to-rectangle

Table 3.2. An example of traffic matrix

	1	2	3	4	5
1	-	15	10	0	1
2	15	-	5	0	0
3	10	5	-	1	0
4	0	0	1	-	20
5	1	1	0	20	-

3.2 General Description of FLPs

In general, non-identical FLPs have different constraints than identical FLPs. So, in this section, I will mention some constraints of FLPs which are often described in previous work, and propose a general evaluation function to evaluate physical layouts. Also, I will suggest a general form of FLP specification.

3.2.1 Constraints of FLPs

A layout can be evaluated from many aspects. Though traffic costs should be mainly considered in FLPs, some other constraints may be also important. Here, I will describe some of them with showing brief examples.

Traffic Matrix Traffic matrix is an essential specification for non-identical FLPs as well as identical ones. The matrix usually consists of $M \times M$ matrix and its i-th row of j-th column means the traffic frequency from the i-th facility to the j-th, where M is the number of facilities. For example, if the traffic matrix is given as Table 3.2, the traffic in a certain period between facilities No.1 and No.2 is fifteen times as that between No.1 and No.5, there is no traffic at all between facilities No.1 and No.4, and so on. (N.B. Although the diagonal elements are shown as '-' for clarity in the table, they can be taken to be 0.)

In Table 3.2, the traffic matrix is symmetric. But, it is not necessarily so. For instance, the traffic may be one directional between workshops in a facility. However, even in such cases, the average traffic of both directions are usually regarded as the traffic between them.

Area Specifications Non-identical FLPs have area specifications for each facility unlike identical FLPs. Usually, the area specification represents the minimum area required by the particular facility. For instance, if this is given as "`3 4 3 2 6`" in a five-facility layout problem, the facilities No.1 and No.3 require areas of the same size, the facility No.2 needs twice as much minimum area as the facility No.4, and so on.

Aspect Ratio Limitations In addition to the area specification, non-identical FLPs usually have aspect ratio limitations, which stand for the acceptable range of the height to width ratio of each facility. For example, if a particular facility has the limitation of 0.75 to 5 and if it requires $12m^2$, this facility should be allocated so that the following limitation can be satisfied.

$$h \times w \geq 12$$

$$0.75 \leq h/w \leq 5$$

where h is the facility height and w is its width.

Orientation Limitations Related to the aspect ratio limitations, some facilities may have orientation limitations, which concerns whether the direction of a particular facility can be rotated 90 degrees clockwise (or counterclockwise). For instance, if there are no orientation limitation in the above example, the acceptable range of the height h and the width w will be changed as follows.

$$h \times w \geq 12$$

$$0.75 \leq h/w \leq 5 \; or \; 0.75 \leq w/h \leq 5$$
$$(i.e. \; 0.2 \leq h/w \leq 5)$$

So, if a facility has *free* orientation limitation (i.e. no orientation limitation), it will be more freely located than the case it has *rigid* orientation limitation. And, maybe this will make the FLP easier.

At that, the free orientation does not mean the complete freedom of the orientation. e.g. An orientation of 38 degrees is not permitted. That is, the FLPs usually assume that each facility is rectangular and to be located horizontally or vertically.

Distance Measurement Methods As already mentioned in the previous chapter, there are two types of methods to measure the distance of two facilities. They are *rectangular* (or *Manhattan*) and *straight* (or *Euclidean*). Whereas the former is the sum of the vertical and horizontal distances between the locations, the latter gets the distance straightforward (geometrically).

For example, if two facilities are located at (4.0, 9.0) and (8.0, 6.0), then the rectangular distance is 7.0 ($because \mid 4.0 - 8.0 \mid + \mid 9.0 - 6.0 \mid = 4.0 + 3.0 = 7.0$) and the straight distance is 5.0 ($because \sqrt{(4.0 - 8.0)^2 + (9.0 - 6.0)^2} = 5.0$).

Incidentally, both of the two methods are used in the standard FLPs which will be introduced in Section 3.3, and the distance is measured from the centre of gravity of a facility.

Room Specifications Room specifications express limitations of the space in which all the facilities should be located. For instance, if there is $100m \times 100m$ space in a factory, and if all the facilities must be assigned into the space, the room specification of $100m \times 100m$ will be given to the FLP.

In addition to the room specification, some prespecified areas may be included in an FLP. For instance, if the room is a non-rectangular shape and/or if the room includes objects like pillars, utilities, etc. which prevents facilities from being put, this specification will be necessary (See Figure 2.7(b)). At that time, the position of each prespecified area should be given with the room specification (See Figure 3.2).

On the other hand, some FLPs may not have such room limitations. In such cases, it may be reasonable to put all facilities as compactly as possible; therefore, the minimum rectangular area involving all the facilities is usually taken into account.

3.2.2 Evaluation Function

FLP's evaluation function for a layout is basically defined by the following formula. And, the aim is to minimise F.

$$F = \sum_{i=1}^{M} \sum_{j=1}^{M} T_{ij} \times D_{f(i)f(j)}$$

where M = the number of facilities,
T_{ij} = the traffic between facilities i and j, and
D_{kl} = the distance between locations of k and l
$f(i)$ = the location of facility i.

However, in addition to the traffic costs, other standards should be also taken into account in non-identical FLPs. So, in order to balance other constraints, the following Formula (3.1) may be reasonable. Actually, this formula covers every evaluation function of the six papers in Table 3.1. In this formula, three terms take account of: the traffic cost; the aspect ratio limitations and the orientation limitations; and the room specifications, respectively.

$$F = Pa \times \sum_{i=1}^{M} \sum_{j=1}^{M} (T_{ij})^a \times (D_{f(i)f(j)})^b + Pb \times \sum_{i=1}^{M} (asp\_break)_i + Pc \times (total\_area) \tag{3.1}$$

where

$M =$	the number of facilities,
$T_{ij} =$	the traffic between facilities i and j,
$D_{kl} =$	the distance between locations of k and l,
$f(i) =$	the location of facility i.
$(asp\_break)_i =$	the degree of to what extent the i-th facility breaks the given aspect ratio limitation,
$(total\_area) =$	the minimum rectangular area that encloses all facilities, and
$a, b, Pa, Pb, Pc =$	appropriate positive numbers, expressing penalty weights.

Choice of penalty values should be straight-forward. E.g. Pc may be either 0 or 1, according to whether or not there is a fixed boundary area. Penalty a is rarely other than 1. Penalty b is sometimes higher than 1, reflecting how high-traffic over long distance is very bad in some FLPs. Pa is sometimes different from 1 to reflect a balance of importance between traffic cost and other factors. Finally, Pb is usually a very large arbitrary number to reflect the fact that layouts breaking an aspect ratio limitation are not valid. Incidentally, the penalty values for each standard FLP are obtained from previous papers as shown in Appendix A.

As for the violation of aspect ratio limitations, [Tam92a], [Tam92b], etc. suggested that the above $(asp\_break)_i$ can be calculated by Formula (3.2) with each facility's aspect ratio limitations, $(asp\_lower\_limit)_i$ and $(asp\_upper\_limit)_i$.

$$(asp\_break)_i = \max\left[0,\ L_i - A_i,\ A_i - U_i\right] \tag{3.2}$$

where

$$\begin{aligned} A_i &= \text{height to width ratio of the } i\text{-th facility} \\ L_i &= \begin{cases} (asp\_lower\_limit)_i & \text{if orientation limitation is rigid} \\ \min\left[(asp\_lower\_limit)_i,\ 1/(asp\_upper\_limit)_i\right] & \text{otherwise} \end{cases} \\ H_i &= \begin{cases} (asp\_upper\_limit)_i & \text{if orientation limitation is rigid} \\ \min\left[(asp\_upper\_limit)_i,\ 1/(asp\_upper\_limit)_i\right] & \text{otherwise} \end{cases} \end{aligned}$$

That is, if A_i is within the limits, $(asp\_break)_i$ will be 0; otherwise, it will be the distance to the nearer end of the allowed range.

However, I would like to claim that this is unreasonable, although this definition looks reasonable. This is because the above definition gives unfairly heavy penalty value to high aspect ratio facilities.

For instance, suppose that there is a facility whose necessary area is 4 and that its aspect ratio limitation is 1 (i.e. $(asp\_lower\_limit) = (asp\_upper\_limit) = 1$). Although the ideal area allocation is 2×2, this facility may be assigned to a non-square area such as 1×4 or 4×1. At that time, because these two non-square areas have the same shape virtually, the penalty for each case should be same. But, Formula (3.2) gives different penalties. That is, as regards the former allocation of 1×4, the aspect ratio is $0.25 (= 1/4)$ and its penalty is $0.75 (= 1 - 0.25)$. On the other hand, the aspect ratio of the latter is $4 (= 4/1)$ and its penalty is $3 (= 4 - 1)$.

Therefore, in order to get a fair penalty value, I would like to use Formula (3.3)

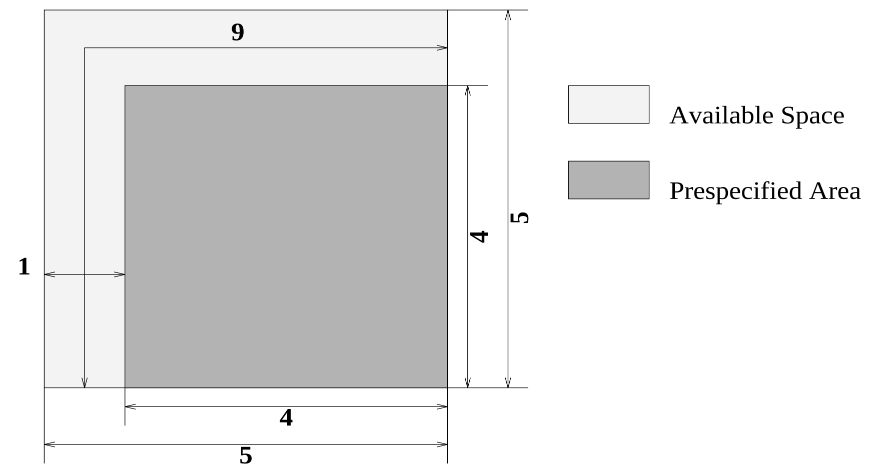


Figure 3.1. A non-rectangular region

below rather than Formula (3.2) in my thesis. By this formula, the rectangles 1×4 and 4×1 in the example have the same penalty value of 3.

$$(asp\_break)_i = \max[0,\ L_i/A_i - 1,\ A_i/U_i - 1] \tag{3.3}$$

where the definitions of A_i, L_i and H_i are same as above.

However, we should recognise that this modified Formula (3.3) still has a limitation because the definition of $(asp\_break)_i$ may become unreasonable in case the available space has a non-rectangular shape related to prespecified areas. For example, let us consider the case shown in Figure 3.1, where the available space has an L-shape due to a prespecified area.

According to [Tam92a] and [Tam92b], the area of a available space is calculated by excluding the prespecified areas and its aspect ratio is regarded as the ratio of the vertical length to the horizontal length of the minimum rectangular region that encloses the available space. Therefore, in this case, the available area and the aspect ratio are calculated as 9 (i.e. $5 \times 5 - 4 \times 4$) and 1 (i.e. 5/5), respectively. Although these values (area = 9 and aspect ratio = 1) seems to indicate it is possible to put a facility requiring area of 9 and a square region into the space, it is actually impossible to put. Hence, this definition may become unreasonable in such cases.

To cope with this problem, many methods are possible. For instance, to obtain more plausible aspect ratio of a non-rectangular region, it can be calculated by the following formula.

$$(aspect\_ratio) = (the\_maximum\_length)/(the\_width\_at\_the\_thinnest\_point)$$

Here, $(the\_maximum\_length)$ means the maximum length of a flexible (bendy) object which can be fit in the region; and $(the\_width\_at\_the\_thinnest\_point)$ means the distance between two sides at the thinnest point of the region. For example, in the case shown in Figure 3.1, $(the\_maximum\_length)$ may be 9, $(the\_width\_at\_the\_thinnest\_point)$ may be 1, and $(aspect\_ratio)$ may be therefore 9 $(= 9/1)$. So, this $(aspect\_ratio)$ can indicate that this space is not suitable for the facility requiring a square region with area of 9. Nevertheless, there will be further arguments. This is because this $(aspect\_ratio)$ cannot distinguish the case shown in Figure 3.1 from a case where the available space is a simple rectangular of 1×9, and because the definition of $(the\_maximum\_length)$ and $(the\_width\_at\_the\_thinnest\_point)$ will be ambiguous when the shape of the available space is more complicated.

As another approach, adding other penalty factors to Formula (3.1) may be a possible idea. For example, if a particular facility is assigned to an L-shape region, a certain penalty can be given to the layout. But, it will be still unclear how much penalty should be given.

In conclusion, there will be more possible approaches which may lead to a lot of arguments. Accordingly, I will not consider this issue here.

3.2.3 A General Expression of FLP specification

As an example of general expression of FLP specification, I would like to suggest a format like Figure 3.2 which is taken from my actual implementation.

Here, `@number` means the number of facilities in this FLP; `@traffic` shows the traffic matrix; `@area` indicates the required area for each facility; `@aspect` specifies the lower and upper aspect limitation as well as the orientation limitation (i.e. `free` or `rigid`); and `@distance_measure` expresses one of the two distance measurement methods (i.e. `manhattan` or `euclidian`).

```
# Tam92-20a

@number
20

@traffic
  0  0  5  0  5  2 10  3  1  5  5  5  0  0  5  4  4  0  0  1
  0  0  3 10  5  1  5  1  2  4  2  5  0 10 10  3  0  5 10  5
  5  3  0  2  0  5  2  4  4  5  0  0  0  5  1  0  0  5  0  0
  0 10  2  0  1  0  5  2  1  0 10  2  2  0  2  1  5  2  5  5
  5  5  0  1  0  5  6  5  2  5  2  0  5  1  1  1  5  2  5  1
  2  1  5  0  5  0  5  2  1  6  0  0 10  0  2  0  1  0  1  5
 10  5  2  5  6  5  0  0  0  0  5 10  2  2  5  1  2  1  0 10
  3  1  4  2  5  2  0  0  1  1 10 10  2  0 10  2  5  2  2 10
  1  2  4  1  2  1  0  1  0  2  0  3  5  5  0  5  0  0  0  2
  5  4  5  0  5  6  0  1  2  0  5  5  0  5  1  0  0  5  5  2
  5  2  0 10  2  0  5 10  0  5  0  5  2  5  1 10  0  2  2  5
  5  5  0  2  0  0 10 10  3  5  5  0  2 10  5  0  1  1  2  5
  0  0  0  2  5 10  2  2  5  0  2  2  0  2  2  1  0  0  0  5
  0 10  5  0  1  0  2  0  5  5  5 10  2  0  5  5  1  5  5  0
  5 10  1  2  1  2  5 10  0  1  1  5  2  5  0  3  0  5 10 10
  4  3  0  1  1  0  1  2  5  0 10  0  1  5  3  0  0  0  2  0
  4  0  0  5  5  1  2  5  0  0  0  1  0  1  0  0  0  5  2  0
  0  5  5  2  2  0  1  2  0  5  2  1  0  5  5  0  5  0  1  1
  0 10  0  5  5  1  0  2  0  5  2  2  0  5 10  2  2  1  0  6
  1  5  0  5  1  5 10 10  2  2  5  5  5  0 10  0  0  1  6  0

@area
100
80
50
60
120
40
20
40
150
120
50
10
20
30
50
20
40
20
80
100

@aspect
0.7     1       free
1       1       free
0.7     1.3     free
0.5     0.8     free
0.9     1       free
0.6     1       free
0.7     1.4     free
1       1       free
0.8     1.1     free
0.5     1.5     free
0.7     1.1     free
0.8     1.2     free
0.95    1.5     free
0.75    1.25    free
0.9     1.1     free
0.8     1.5     free
0.4     1.4     free
0.9     1.9     free
1       1       free
0.95    1.15    free

@distance_measure
manhattan

@eval_method
TxDx2plus100xASP_ratio

@room
0 0 40 35
@objects
4
0 0 10 5
30 30 40 35
0 30 10 35
20 20 30 25
```

Figure 3.2. A sample of FLP specification

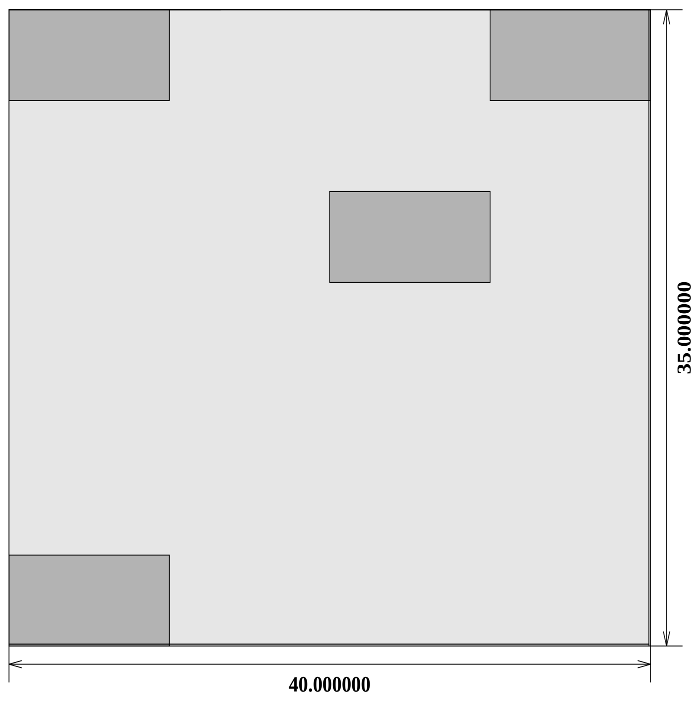


Figure 3.3. An example of a room specification

In addition, in my implementation, the evaluation function is symbolically indicated at `@eval_method` section. So, in this case, "`TxDx2plus100xASP_ratio`" means Formula (3.1) with $a = 1$, $b = 1$, $Pa = 2$, $Pb = 100$, and $Pc = 0$, and Formula (3.3) will be used.

As regards the room specification, the sections `@room` and `@objects` give the information. In this example, "`0 0 40 35`" at `@room` section shows the room's left lower corner is at (0,0) and its right upper corner is at (40, 35). Similarly, `@objects` section expresses the prespecified areas' information. While the next line of `@objects` indicates the number of prespecified areas, four sequential numbers in lower lines indicate each object's position. So, in this case, the room area will be like Figure 3.3.

3.3 Fifteen Standard Problems

In this section, I will describe the FLPs chosen as the standard problems from Table 3.1. Some figures are shown here in order to give the reader a brief image.

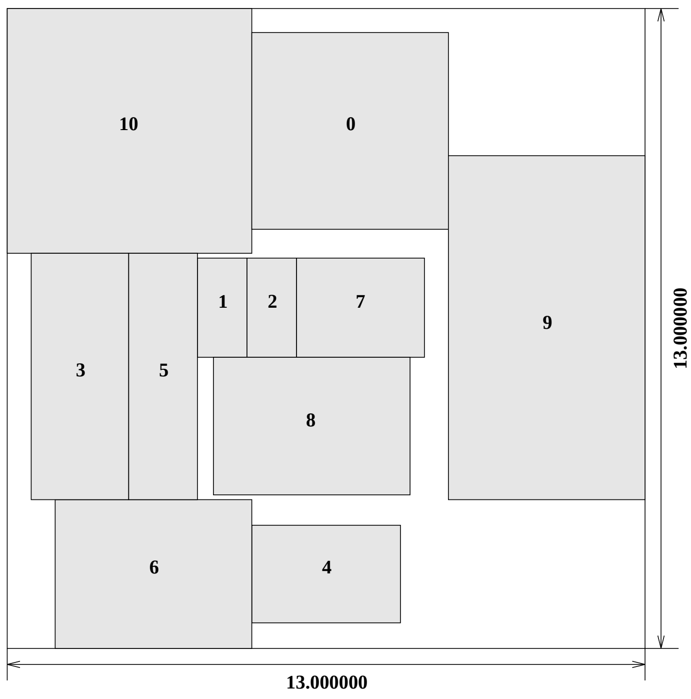


Figure 3.4. A good layout of Kea91-11

The implementation of these problems can be seen in Appendix A in the format of Figure 3.2.

3.3.1 The description of fifteen standard problems

Kea91-11† (⋆1 of Table 3.1)

Although this problem originally described in [IM89], it did not include any quantitative results. Later, [KJK91] studied this problem and reported a good physical layout with its evaluation function and the score. The good layout can be generated by STS, and I can confirm that the layout shown in Figure 3.4 has the same score of 2829.4.

Kea91-11a† (⋆2 of Table 3.1)

This problem is same as Kea91-11 except that the aspect ratio limitation is soft. That is, whereas each facility in Kea91-11 should have a particular aspect ratio, each facility in Kea91-11a can have any aspect ratio between 0.25 and 4

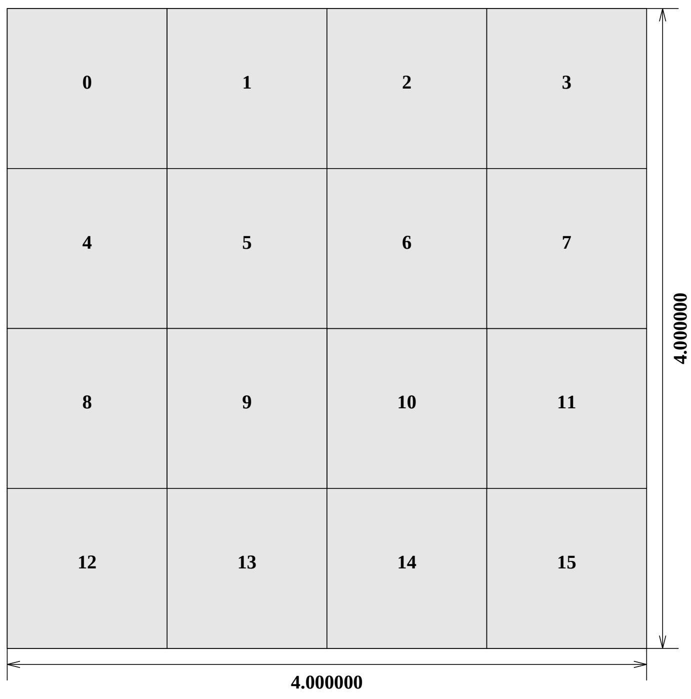


Figure 3.5. An ideal layout of Kea91-16

without penalty. A good layout was also reported by [Kea91-11] with its score of 2287.041.

Kea91-16† (⋆3 of Table 3.1)

This problem was initially suggested and solved by [CHMR91] and later compared by [KJK91]. This artificially created problem contains sixteen identical square facilities with rather small traffic, and this has the ideal layout shown in Figure 3.5. This ideal layout could be produced by the [KJK91]'s method and I can confirm its score is 64.

Kea91-20† ‡ (⋆4 of Table 3.1)

This problem was also first created by [CHMR91] and compared by [KJK91]. Though this problem was similarly created artificially, this includes non-identical facilities and the traffic matrix is a little complicated. According to [KJK91], this problem's ideal layout shown in Figure 3.6 should have the score of 125. However, this layout has not been reached by any methods so far.

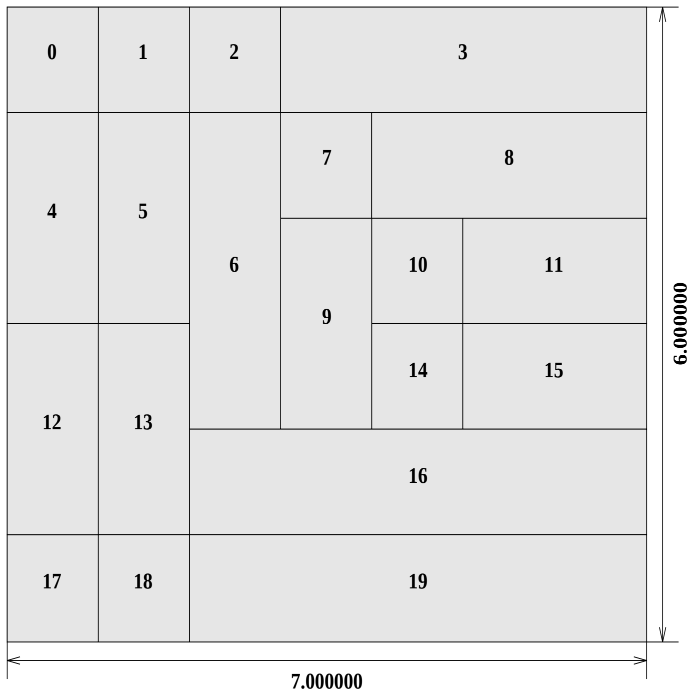


Figure 3.6. An ideal layout of Kea91-20

TL91-5†, TL91-6†, TL91-7†, TL91-8†, TL91-12†, TL91-15† (⋆5 - ⋆10 of Table 3.1, respectively)

[TL91] created these problems by adding area specification and aspect ratio limitations to [NVR68]'s FLPs. Because the physical layouts generated by [TL91] can not be represented by STS, I was not able to confirm the scores of them.

TL91-20†, TL91-30† (⋆11 and ⋆12 of Table 3.1)

Similarly, [TL91] created these problems based on [NVR68]'s corresponding problems. Nevertheless, these problems could not be solved by [TL91] directly. Instead, [TL91] suggested that these big size problems could be solved separately and they might be merged later.

Tam92-20a∗, Tam92-30a∗ (⋆13 and ⋆14 of Table 3.1)

As well as [TL91], [Tam92b] produced these problems based on [NVR68]'s problems. Because [Tam92b]'s specifications are different from [TL91], these became different problems. Unlike [TL91], [Tam92b] put room specification with

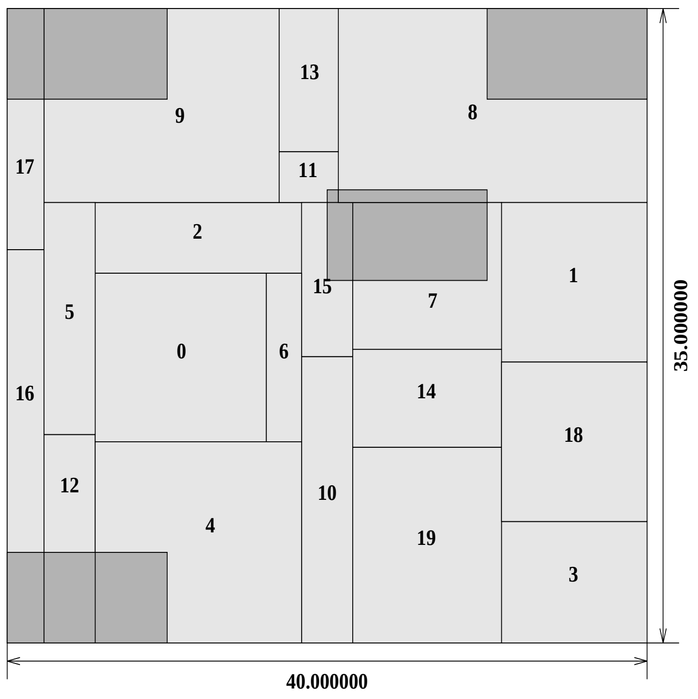


Figure 3.7. A good layout of Tam92-20a

prespecified areas into these problems. Although the best layout among ten experiments are shown in [Tam92b], I was not able to confirm these scores. That is, the scores of the layouts shown in Figure 3.7 and Figure 3.8 were 25779.53 and 47422.3 according to [Tam92b]; nevertheless, they became 24389.24 and 45104.4 in my calculation. Though I made efforts to find out the cause of the difference, I have not found out the reason yet. There might be some typographical errors in FLP specifications in the paper.

VCea91-10† (⋆15 of Table 3.1)

This problem is suggested and solved by [VCCV91]. A good layout, whose score is 24445, was also reported; however, the same layout shown in Figure 3.9 showed the score of 24152 by my calculation. Although I looked for the cause carefully, it has not been caught. Consequently, considering the difference is less than 1.5%, I would like to think it might be a rounding-off error.

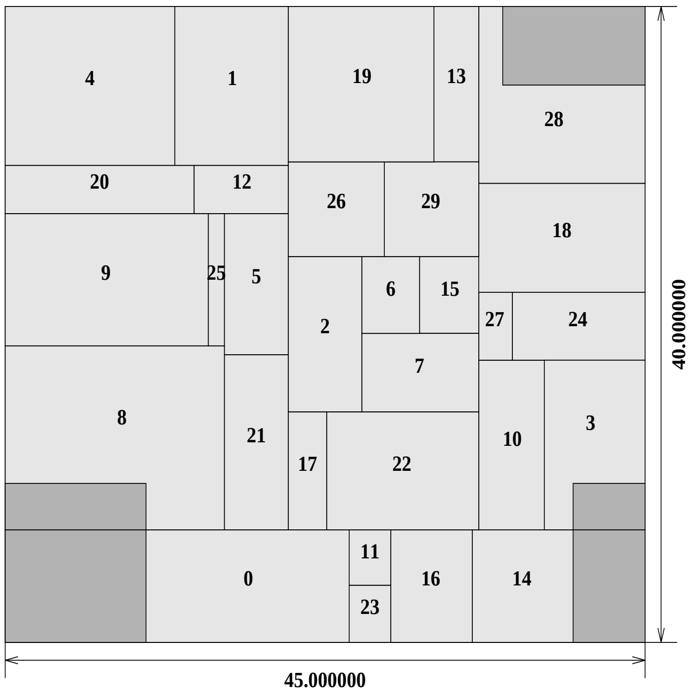


Figure 3.8. A good layout of Tam92-30a

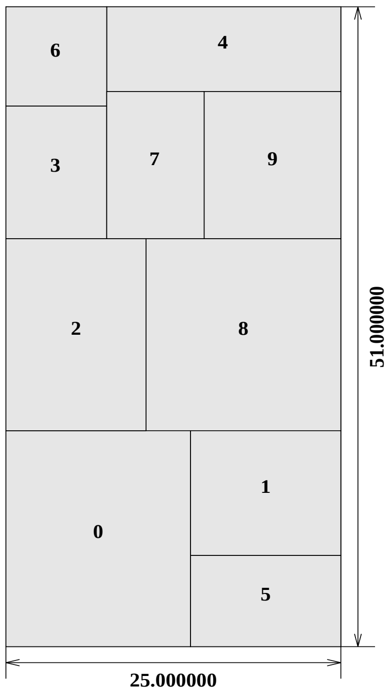


Figure 3.9. A good layout of VCea91-10

3.3.2 Some Amendments

† **Auxiliary Penalty Function** Except for Tam92-20a and Tam92-30a, all FLPs did not add penalty for the violation of aspect ratio limitations. That is, Pb in the Formula (3.1) was 0 in the FLPs. However, because the FLPs gave each facility some aspect ratio limitation, the penalty weight Pb should be assumed as a positive number in case some facilities break their aspect ratio limitations. Accordingly, I modified these problems' evaluation functions by putting the value of 1000000 in Pb of Formula (3.1).

‡ **Amendment for Kea91-20** As for this problem, I suspect the both traffic matrices shown in [CHMR91] and in [KJK91] may be wrong. Both of the papers clearly mentioned that the traffic cost of the ideal layout shown in Figure 3.6 is 83; nevertheless, it became 127 and 85 by the traffic matrices shown in [CHMR91] and in [KJK91], respectively, when I calculated it carefully. Therefore, I removed the traffic between the facilities No.10 and No.14 from the traffic matrix in [KJK91] so that its ideal layout can have the traffic cost of 83.

$*$ **Amendment for Tam92-20a and Tam92-30a** Though the evaluation function in [Tam92b] includes the aspect ratio penalty, the definition in Formula (3.2) will be unreasonable. That is, under this unreasonable penalty, clearly worse layouts might be regarded as better layouts. Hence, I used Formula (3.3) instead of Formula (3.2) and would like to distinguish these modified problems from originals by calling them Tam92-20a and Tam92-30a rather than Tam92-20 and Tam92-30. The corresponding scores of the best layouts reported in [Tam92b] are now 23544 and 45044 under this new evaluation function.

Chapter 4

The Slicing Tree Structure (STS)

4.1 What is Slicing Tree Structure?

The slicing tree structure was originally described in [Ott82]. He suggested that many layouts could be expressed by slicing stages of top-down cuttings (See Figure 4.1). As shown in Figure 4.2(a) and (b), this top-down cutting stages can be also represented by a tree, where each terminal node corresponds to a facility and each non-terminal node means the relative position of facilities. Here, the numbers in the terminal nodes express the facility's index and the letters in the non-terminal nodes express the relation of each substructure, where the relation is one of the following two:

`+`	=	*"The substructure given by the second argument is just above the substructure given by the first argument."*
`*`	=	*"The substructure given by the second argument is just left of the substructure given by the first argument."*

Also, an STS can be expressed by a Polish expression, where terminal nodes and non-terminal nodes are considered as operands and operators, respectively. That is, the STS in Figure 4.2(b) can be represented by a Polish expression of `123+*4*`. These sorts of trees are called STSs.

As [WL86] mentioned, this STS introduces a redundancy of representation because a particular layout may be expressed by some different STSs or Polish expressions. For example, the layout shown in Figure 4.2(a) can be also represented by the STS shown in Figure 4.2(c) because the layout has two vertical

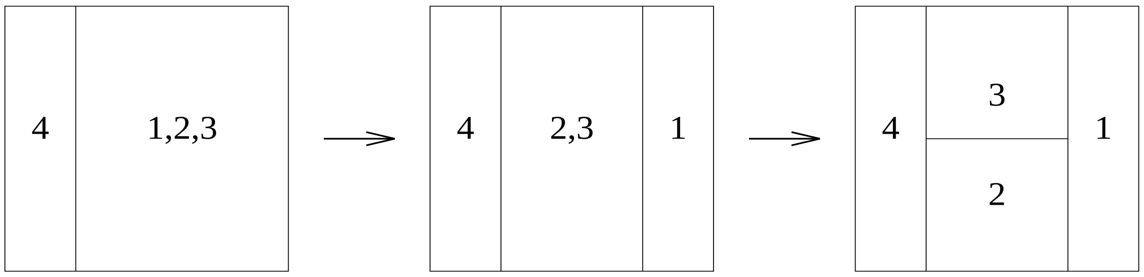


Figure 4.1. Cutting stages for a layout

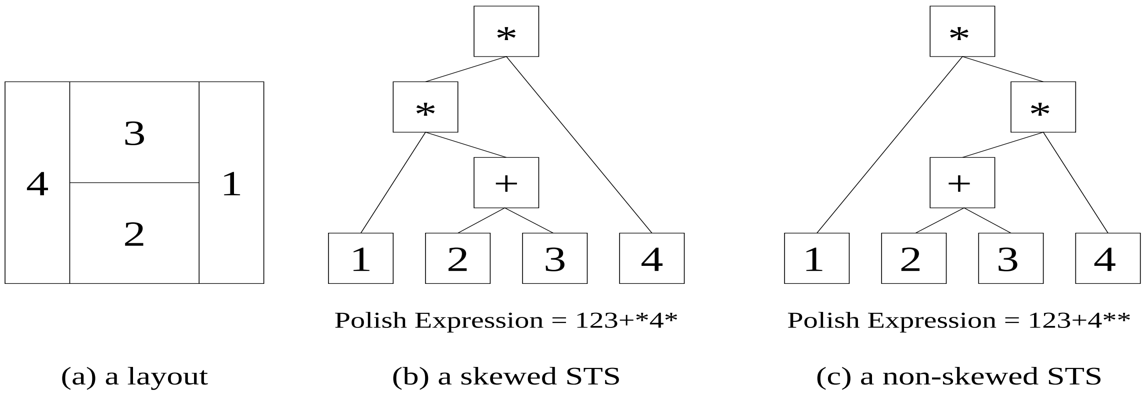


Figure 4.2. An example of Slicing Tree Structures

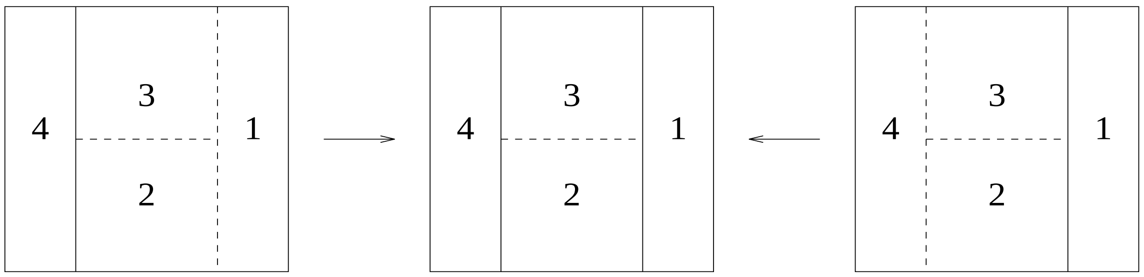


Figure 4.3. Alternative cutting ways for a layout

cuts which can be done in any order. (See Figure 4.3). So, if a layout includes some direction's cuts which can be done in any order, in a particular stage, there must be more than one STS corresponding to the layout.

In order to remove this redundancy, [WL86] suggested a rule as follows: *"A non-terminal node's right-hand-side child node must be either a non-terminal node having the other operator than that of the parent, or a terminal node."* In other words, this rule can be stated as follows: *"The Polish expression corresponding to an STS must not have the same operators in adjacent positions."* For example, the Polish expression `123+4**`, which expresses the STS shown in Fig 4.2(c), breaks this rule in the part of `**`. [WL86] called the STSs and Polish expressions satisfying this rule, skewed STSs and normalized Polish expressions, respectively. So, the STS shown in Figure 4.2(b) is a skewed STS, whereas that shown in Figure 4.2(c) is not.

Indeed, a canonical perspective might support [WL86]'s rule. As [WL86] mentioned, simulated annealing for an FLP using only skewed STSs might get better results than that using arbitrary STSs because of the smaller search space. However, [CHMR91] reported the opposite result that redundant representation led better performance. Also, they commented that it is probably because [WL86]'s rule makes STS's usage much complicated and restricted. So, I cannot help being doubtful about [WL86]'s suggestion; consequently, I decided to decline [WL86]'s rule in my research.

Incidentally, [Tam92a] and [Tam92b] suggested STSs with four types of operators as follows. Although it more increases STS redundancy, this kind of redundancy (i.e. the redundancy between `U/L` and `B/R`) is not always useless because the room is sometimes not symmetric due to prespecified areas.

`U`	=	*"The substructure given by the second argument is just above the substructure given by the first argument."*
`B`	=	*"The substructure given by the second argument is just beneath the substructure given by the first argument."*
`L`	=	*"The substructure given by the second argument is just left of the substructure given by the first argument."*
`R`	=	*"The substructure given by the second argument is just right of the substructure given by the first argument."*

Merits of STSs As already mentioned in Section 2.1, there are some layouts which can be expressed by the STS but which can not be expressed by other methods such as the cell assignment method and the flexible bay structure. For instance, the cell assignment method, which is one of the most common representations for identical FLPs, generally creates shapes too strange for practical use as shown in Figure 2.2 in non-identical FLPs. On the other hand, complicated layouts as shown in Figure 2.5(a) can not be expressed by the flexible bay structure. Also, because Polish expressions may be a suitable representation for a computation, this may be helpful for the implementation.

Limitations of STSs However, there are some limitations in STSs. First, there are some physical layouts which cannot be represented by the STS. For example, the layout shown in Figure 4.4 is impossible to be represented by STSs, though it may be a quite natural layout. That is, STSs can only express layouts which can be separated in two rectangular subparts recursively.

Second, STSs may not be a suitable representation for some FLPs. For example, if the room is given as shown in Figure 4.5(a) and if a room is required to be separated into three approximately equal areas, the reasonable layout may be as shown in Figure 4.5(b). Nevertheless, STS can represent only poor layouts such as Figure 4.5(c) because the cuts represented by STSs must follow the horizontal or vertical direction.

Therefore, we should recognise that the STS is not ideal in every case, though the STS may be a good representation in many cases.

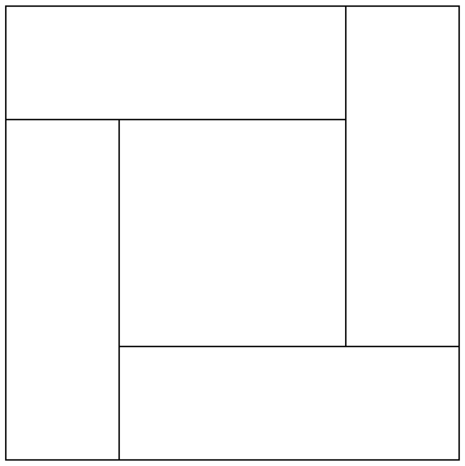


Figure 4.4. An impossible layout by STSs

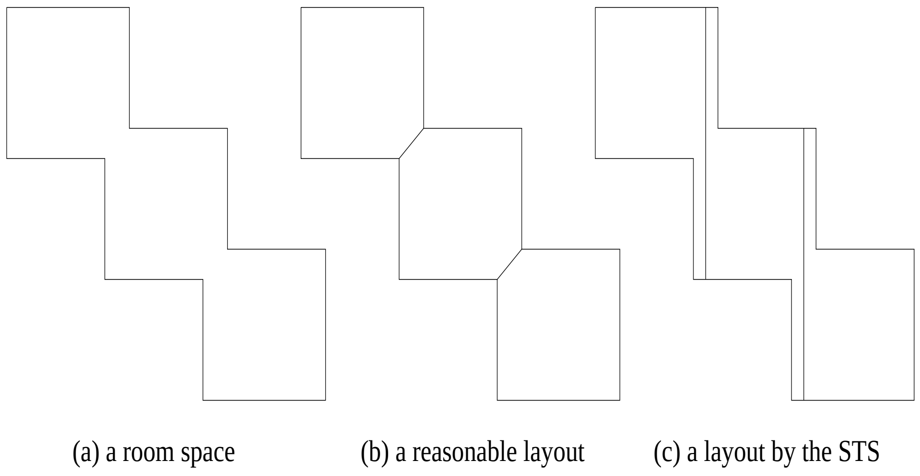


Figure 4.5. A difficult FLP for the STS

Table 4.1. An example of FLP specifications

facility No.	required area	aspect ratio limitation	orientation limitation
1	100	0.5 - 0.8	free
2	200	0.95 - 1.25	free
3	300	0.8 - 1.0	free
4	300	1.0 - 1.0	free

4.2 The Top-down Interpretation

[Tam92a] and [Tam92b] suggested a method of generating a physical layout from an STS for an FLP with room specification. Here, I will call it "Top-down Interpretation" because this method reads the STS from the top node to the bottom nodes. The procedure is as follows.

First, the available region (the room area excluding prespecified areas) is divided into two sub-regions. Then, each sub-region is divided again into two sub-sub-regions. By repeating this procedure, the whole region will be divided into small regions so that one region can correspond to one facility. For instance, as shown in Figure 4.6(a), the room is first divided into sub-regions for facility group 1 and 2 and for facility group 3 and 4, respectively. After that, two sub-regions are further divided into four sub-sub-regions corresponding to each facility. At that time, the dividing line's position is decided by the ratio of one facility group's required area to another.

To explain this method, I will show an example. First, suppose that an STS and each facility's required area are given as shown in Figure 4.7(a) and Table 4.1. Also, we assume that the facilities should be located in 30×30 room area.

Because the facilities No.1 and No.2 require the area of 300 altogether and the facilities No.3 and No.4 require the area of 600 altogether, the first cut should be done so that two sub-regions area ratio can be one to two, as shown in Figure 4.7(b).

Similarly, considering that the required area ratios between facility No.1 and No.2 and between No.3 and No.4 are one to two and one to one, the final layout

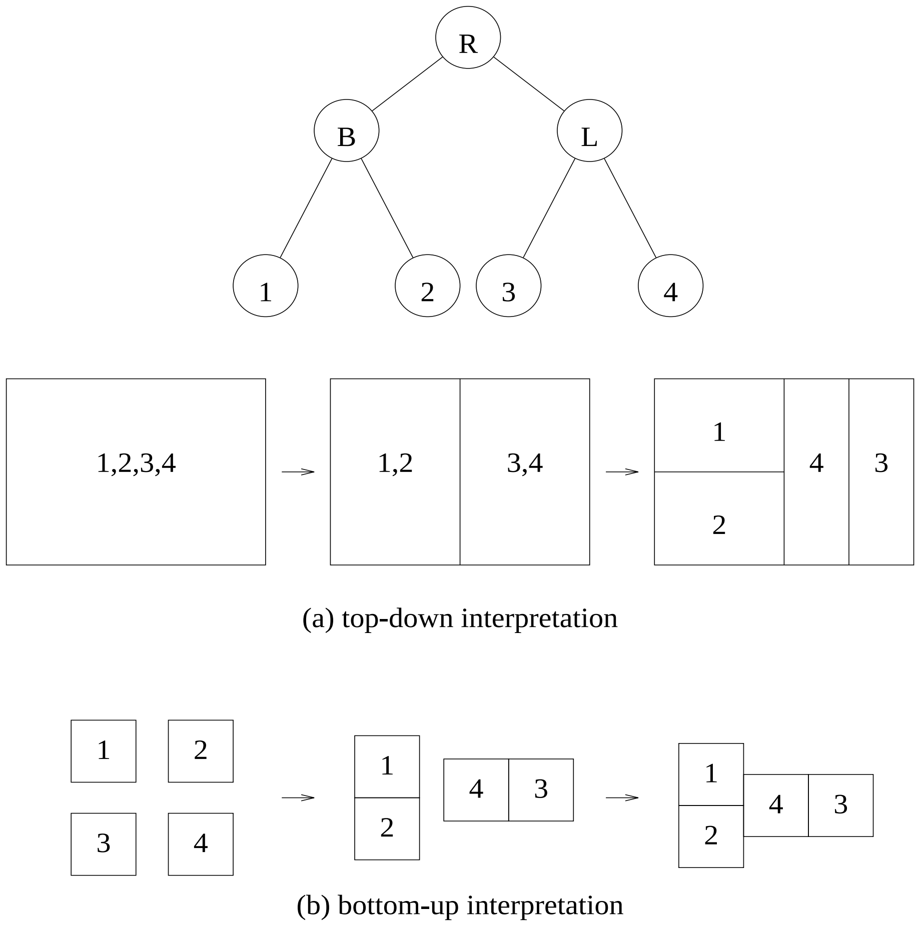
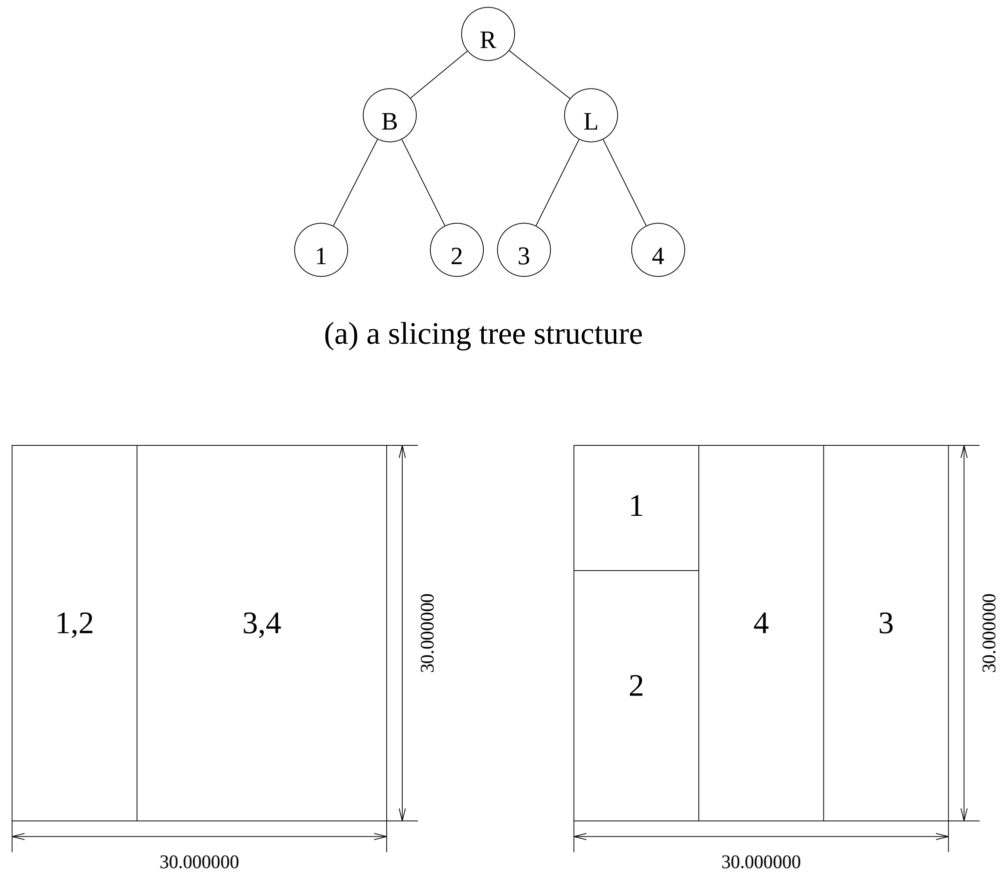


Figure 4.6. How to translate STS to geometry



(a) a slicing tree structure

(b) the layout after the first cut

(c) the final layout

Figure 4.7. The top-down interpretation

will be generated as shown in Figure 4.7(c).

Of course, the layout in Figure 4.7(c) will have low fitness due to the violation of the aspect ratio limitations by all the facilities. But this is an example to show how the "Top-down Interpretation" can be done.

4.3 The Bottom-up Interpretation

In the FLPs without room specifications, the physical layout corresponding to an STS may be built by some sort of bottom-up approach. For example, as shown in Figure 4.6(b), each facility can be connected from bottom nodes of the STS to the top node. However, as [KJK91] mentioned, how to decide the shape of each facility may be a problem because the facility's shapes greatly influence the produced layouts score. Nevertheless, the method [KJK91] used might be too complicated to be calculated quickly; accordingly, I established my own method which can decide each facility's reasonable shape quickly as follows. For convenience, I will call this method "Bottom-up Interpretation".

```
Step 1:  Assume a square region where its size is equal to the total area
         required by the facilities. And decide a temporary physical
         layout by the top-down interpretation.

Step 2:  Transform the square region to satisfy each facility's
         aspect ratio limitation. At that time, the transformed region
         should be kept as small as possible.

Step 3:  Shrink each facility's assigned area as much as possible.
         The shrunk shapes are used as their final shapes.

Step 4:  Construct a physical layout by the bottom-up manner as
         shown in Figure 4.6(b). At that time, two facilities/groups
         are connected so that the centre of each facility/group
         aligns horizontally or vertically.
```

In order to explain it, I will show an example. First, suppose that FLP specifications and the STS are given as shown in Table 4.1 and Figure 4.8(a).

As Step 1, because the total required area is 900, a 30×30 square room is assumed. Using the top-down interpretation, a temporary layout is decided as shown in Figure 4.8(b).

As Step 2, the aspect ratio limitation of each facility is examined. For example, the facility No.4 needs a square region, but it is assigned to 30×10 space in the temporary assignment; therefore, the shape should be changed. To satisfy the aspect ratio limitations of all facilities, transforming factors H_i and V_i are calculated, where H_i (or V_i) means the scale for horizontal (or vertical) direction's enlargement/reduction for the i-th facility.

From the FLP specifications, the required minimum length and width of each facility can be obtained. Here, suppose that A_i is the i-th facility's required area, B_i and C_i stand for the facility's lower and upper aspect ratio limitation, and L_i and W_i are the facility's length and width.

If we keep the facility's area to be its minimum requirement, the equation below follows.

$$L_i \times W_i = A_i$$

At that time, the following condition should be satisfied.

$$B_i \leq \frac{L_i}{W_i} \leq C_i$$

Because they can be transformed as follows, the minimum values for L_i and W_i can be considered as $\sqrt{A_i \times B_i}$ and $\sqrt{A_i / C_i}$, respectively.

$$\sqrt{A_i \times B_i} \leq L_i \leq \sqrt{A_i \times C_i}\,,\ \sqrt{\frac{A_i}{C_i}} \leq W_i \leq \sqrt{\frac{A_i}{B_i}} \qquad (4.1)$$

Therefore, comparing these minimum values with the length and width of the temporarily assigned area for each facility in Step 1, the transforming factors H_i and V_i can be calculated.

For instance, let us consider the case of facility No.3. As shown in Figure 4.8(b), the facility No.3 is temporarily assigned to 30×10 area. On the other

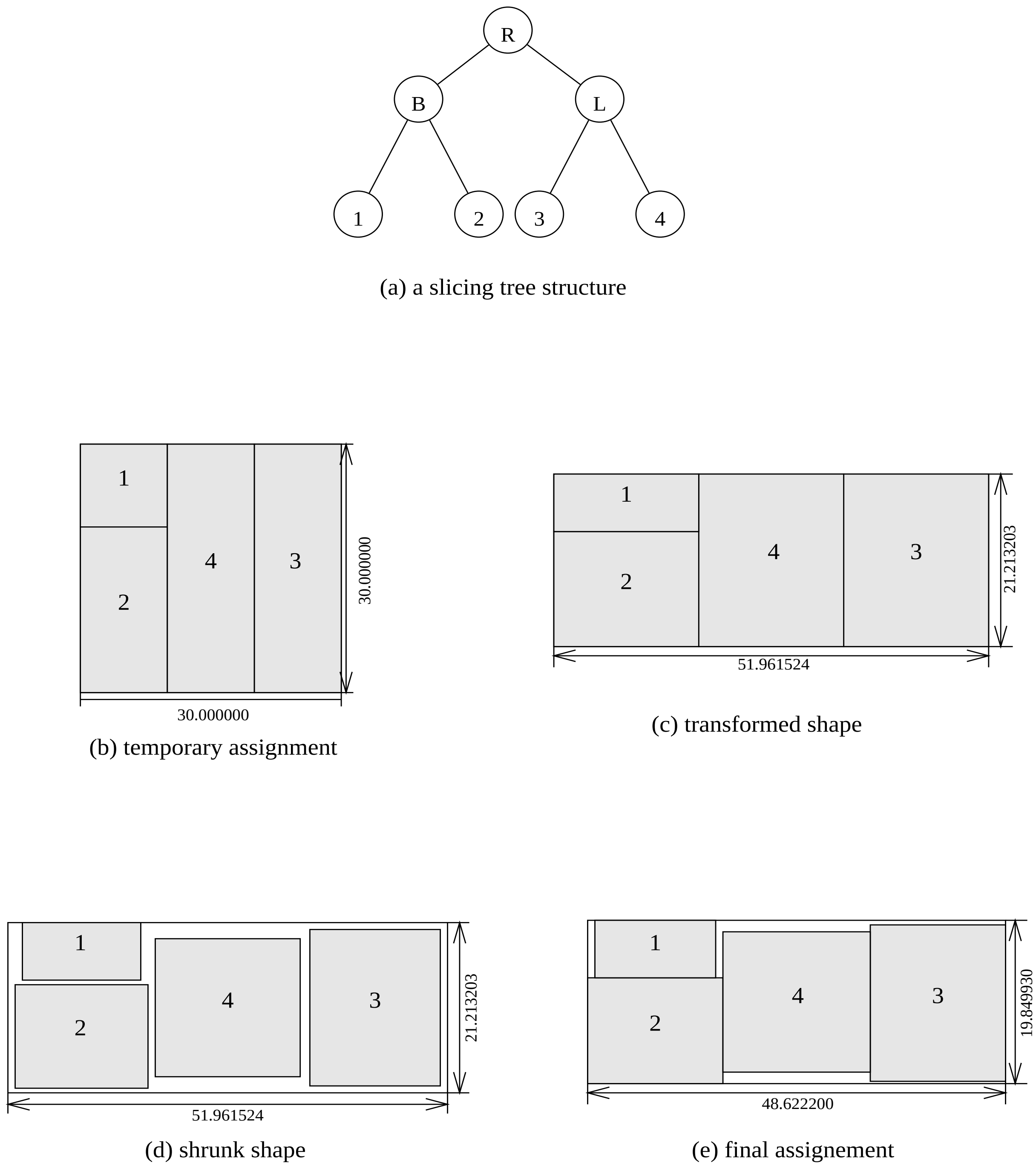


Figure 4.8. The bottom-up interpretation

Table 4.2. Enlarging/reducing factors for each facility

No.	A_i	B_i	C_i	$\sqrt{A_iB_i}$	$\sqrt{A_i/C_i}$	temporary shape (set in Step 1)	V_i	H_i
1	100	0.5	0.8	$\sqrt{50}$	$\sqrt{125}$	10×10	71%	112%
		1.25	2.0	$\sqrt{125}$	$\sqrt{50}$		112%	71%
2	200	0.8	1.25	$\sqrt{160}$	$\sqrt{160}$	20×10	63%	126%
3	300	0.8	1.25	$\sqrt{240}$	$\sqrt{240}$	30×10	52%	155%
4	300	1.0	1.0	$\sqrt{300}$	$\sqrt{300}$	30×10	58%	173%

A_i	=	the required area of the i-th facility
B_i	=	the lower aspect ratio limitation of the i-th facility
C_i	=	the upper aspect ratio limitation of the i-th facility
V_i	=	required vertical enlarging/reducing factor for the i-th facility
H_i	=	required horizontal enlarging/reducing factor for the i-th facility

hand, because its orientation limitation is free (i.e. it can be rotated 90 degrees), its aspect ratio limitation is 0.8 to 1.25 (= 1/0.8). Therefore, by Formula (4.1) the minimum length and width for the facility No.3 is obtained as follows.

$$\sqrt{A_3 \times B_3} = \sqrt{300 \times 0.8} = 15.5 \ , \ \sqrt{\frac{A_3}{C_3}} = \sqrt{\frac{300}{1.25}} = 15.5$$

So, the temporary assignment should be enlarged at least 155% (= 15.5/10) in the horizontal direction, whereas it can be reduced to at most 52% (= 15.5/30) in the vertical direction. That is, $H_3 = 155\%$ and $V_3 = 52\%$.

Applying similar method to every facility, we can get enlarging/reducing factors H_i and V_i as shown in Table 4.2.

In this example, facility No.1 has two permissible ranges of aspect ratio, 0.5 to 0.8 and 1.25 (= 1/0.8) to 2.0 (= 1/0.5). So, if we take the former range, the possible vertical reducing scale for all facilities will be 71% (i.e. max(71%, 63%, 52%, 58%)) and the required horizontal enlarging scale for all facilities will be 173% (i.e. max(112%, 126%, 155%, 173%)) at least. In contrast, if we take the latter range, they will be 112% and 173%, respectively. Because, after the transformation of the temporary area, the former case will need smaller area enclosing all facilities than the latter; the former may be better. Thus, the whole facility areas are transformed as shown in Figure 4.8(c) using the scales 71% and 173%.

As Step 3, the enlarged facility areas are shrunk to reduce redundant areas. This may be possible because Step 2 generally makes every facility's area larger than its requirement.

In this step, if the area can be reduced in both horizontal and vertical directions, the aspect ratio will be kept; otherwise, the area will be reduced as much as possible in either direction. And, the shrunk area's shape will be used as the facility's final shape. In this example, the shrunk shapes are shown in Figure 4.8(d).

As Step 4, the final layout is created by connecting facilities by the bottom up manner as shown in Figure 4.6(b). At that time, two facilities/groups are connected so that the centre of each facility/group aligns horizontally or vertically, based on the STS operator. Incidentally, the centre of a group of facilities is defined as the centre of the minimum rectangular area that encloses all facilities in the group.

For instance, if the STS and the shrunk shapes are given as shown in Figure 4.8(a) and Figure 4.8(d); the facility No.2 is connected to be just beneath No.1, the facility No.4 is connected to be just left of No.3, and the group consisting of No.3 and No.4 is connected to be just right of the group consisting of No.1 and No.2. So, the final layout will be as shown in Figure 4.8(e).

Possible Enhancements As shown in Figure 4.8(e), the final layout may have some small gaps. So, if we push facilities from both directions, the layout may be improved. And, even if there are no gaps between facilities at all, we may be able to continue pushing by transforming each facility's shape. However, for example, pushing may cause an aspect ratio violation again; consequently, I will not consider this issue here.

4.4 STSs and Search Space

In this section, I will introduce how many different layouts can be represented by the STS. Because the number of layout variations specifies the size of search space, this calculation will be valuable. Also, I will introduce reduced Polish expressions which consists of operators only. Because this reduced expression can limit the search space, it may be helpful for FLPs to find good solutions quickly. Finally I will mention how to decode a reduced Polish expression to an ordinary Polish expression with a Polish expression's template.

4.4.1 STS topology

At first, I will consider how many different topologies can be expressed by the STS including N terminal nodes. Here, I will call the number C_N.

To obtain C_N, let us consider the cases where N is a small number. First, if $N = 1$ or 2, it is obvious that $C_1 = C_2 = 1$. Second, if $N = 3$, there are two different topologies as shown in Figure 4.9(a); therefore, $C_3 = 2$. As for $N = 4$, the possible structures must come down one of three groups shown in Figure 4.9(b); therefore, C_4 can be calculated as follows.

$$C_4 = C_1 \cdot C_3 + C_2 \cdot C_2 + C_3 \cdot C_1 = 2 + 1 + 2 = 5$$

Similarly, for $N = 5$, the possible structures must come down one of four groups shown in Figure 4.9(c); therefore, C_5 can be calculated as follows.

$$C_5 = C_1 \cdot C_4 + C_2 \cdot C_3 + C_3 \cdot C_2 + C_4 \cdot C_1 = 5 + 2 + 2 + 5 = 14$$

From these observations, the formula below follows.

$$C_N = \begin{cases} \sum_{i=1}^{N-1} C_i \cdot C_{N-i} & (N \geq 2) \\ 1 & (N = 1) \end{cases} \tag{4.2}$$

Although this is a recurrence formula, the value of C_N can be calculated and C_N are known as Catalan numbers [VLW92]. Here, I will describe how to obtain the value of C_N.

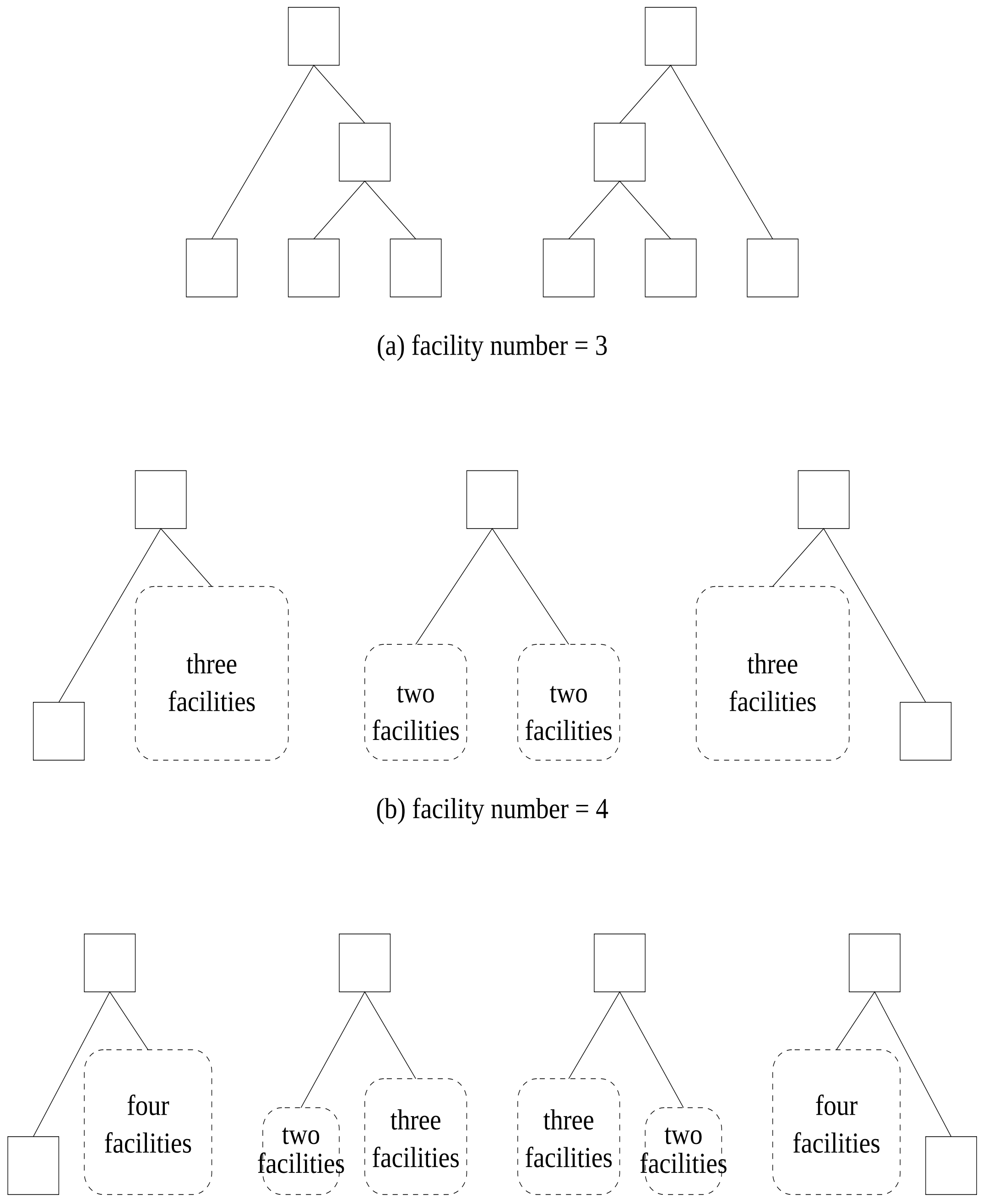


Figure 4.9. Classifications of FLP topology

Suppose that a generating function $c(x)$ is defined as follows.

$$c(x) = \sum_{j=1}^{\infty} C_j x^j$$

Then

$$\begin{aligned} c(x)^2 &= \sum_{k=2}^{\infty}(\sum_{i=1}^{k-1} C_i C_{k-i}) x^k \\ &= \sum_{k=2}^{\infty} C_k x^k \\ &= c(x) - x \end{aligned}$$

This is a quadratic equation in $c(x)$ and so

$$c(x) = \frac{1}{2}(1 \pm \sqrt{1-4x})$$

Since we require that $c(0) = 0$ it is necessary to choose the minus sign. Expanding the square root as a power series therefore gives us:

$$\begin{aligned} C_n &= -\frac{1}{2}\begin{pmatrix} 1/2 \\ n \end{pmatrix}(-4)^n \\ &= -\frac{1}{2}(\frac{1}{2})(\frac{-1}{2})(\frac{-3}{2})(\frac{-5}{2})\cdots(\frac{-(2n-3)}{2})\frac{(-4)^n}{n!} \\ &= \frac{1 \cdot 3 \cdot 5 \cdots (2n-3) \cdot 2^{n-1}}{n!} \\ &= \frac{1 \cdot 3 \cdot 5 \cdots (2n-3) \cdot 2^{n-1}}{n!} \times \frac{2 \cdot 4 \cdot 6 \cdots (2n-2)}{1 \cdot 2 \cdot 3 \cdots (n-1) \cdot 2^{n-1}} \\ &= \frac{(2n-2)!}{(n-1)!n!} \end{aligned}$$

Hence,

$$C_n = \frac{(2n-2)!}{(n-1)!n!} = \frac{1}{n}\begin{pmatrix} 2n-2 \\ n-1 \end{pmatrix} \tag{4.3}$$

4.4.2 The Size of Search Space

An STS expressing N facilities must have N terminal nodes and $(N-1)$ non-terminal nodes. So, whereas the number of variations of terminal nodes is $N!$, that of non-terminal nodes is 4^{N-1} when four types of operator are used. Therefore, the number of variations of STSs for a N-facility problem is

$$C_N \cdot 4^{N-1} \cdot N! = \frac{(2N-2)!}{(N-1)!} \cdot 4^{N-1} \tag{4.4}$$

In other words, we can say that an FLP consisting of N facilities is a problem of searching the best solutions among $(2N-2)! \cdot 4^{N-1}/(N-1)!$ alternatives.

4.4.3 Reduced Polish Expressions and Polish Expression's Templates

As mentioned above, the search space of an FLP including N facilities is generally huge. So, reducing the search space may be a good approach to get better layouts quickly.

[Tam92a] and [Tam92b] introduced an idea of partially fixed STSs as follows. First, a reasonable STS topology is chosen by some means from C_N possible topologies. Second, every terminal node, which corresponds to each facility, is decided by some means from $N!$ possible variations. For example, a partially fixed STS for an FLP consisting of five facilities may be decided as shown in Figure 4.10 where numbers `1` to `5` indicate the facility's indexes and letters `x,y,z,w` mean the places of operators. At that time, the STS can be expressed not only by an ordinary Polish expression, `23z145wyx`, but also by `zwyx` simply, because the STS topology and operands positions are fixed.

That is, if we assume that we do not consider any other STS topologies and any other terminal node variations, an STS can be expressed by a reduced Polish expression which contains operators only. So, in this case `zwyx` is the corresponding reduced Polish expression.

A reduced expression can be decoded to an ordinary Polish expression by using the information of STS topology and operand positions. For instance, the information can be retained as a template such as `23%145%%%`, where `%` means an

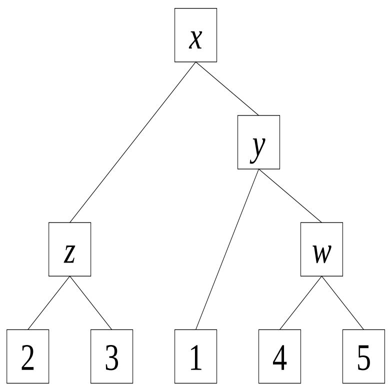


Figure 4.10. A partially fixed STS

operator's position. So, if we obtain a reduced Polish expression, say, `URBL`; by putting each operator into the template in the same order, we can get an ordinary Polish expression `23U145RBL`. In my thesis, I would like to call this template the Polish expression's template.

Because there are only 4^{N-1} variations in the partially fixed STSs expressing N facilities, partially fixed STSs and reduced Polish expressions may be a helpful idea to reduce search space.

4.5 Summary

In this chapter, beginning with the description of STSs, I introduced two types of interpretations, Top-down Interpretation and Bottom-up Interpretation, as a method for corresponding an STS to a physical layout. While Top-down Interpretation is useful for FLPs with room specifications, Bottom-up Interpretation is useful for FLPs without them.

In addition, I calculated how many different layouts can be expressed by the STS consisting of N terminal nodes. The value, $(2N-2)! \cdot 4^{N-1}/(N-1)!$, indicates the size of search space for an FLP consisting of N facilities. And, I introduced partially fixed STSs and reduced Polish expressions which can reduce the search space size as well as how to decode a reduced Polish expression to an ordinary Polish expression.

Chapter 5

Some GAs for FLPs

In this chapter, I will explain six types of GAs I implemented. For convenience, I will call them Cea, Tam, DK, Tam2, DK2, and Kad algorithms. The Cea and Tam algorithms are duplicates of the GAs in [CHMR91] and [Tam92a]. Although the DK algorithm is almost the same as the Tam algorithm, it uses a clustering method introduced by [DK85]; therefore, I called it DK. The Tam2 and DK2 algorithms use different chromosome representations from Tam and DK so that they can widen the search space, though they are similar to Tam and DK, respectively. The Kad algorithm is a hybrid algorithm of Tam2 and DK2.

5.1 The Cea algorithm

Chromosome Representation In the Cea algorithm, the Polish expression corresponding to the STS is directly used as the chromosome representation. However, special crossovers and mutations are necessary because conventional ones may produce invalid Polish expressions. For example, if `12U3B` and `312RL` are one-point crossed over and if the splitting point is chosen between third and fourth genes, the children will be `12URL` and `3123B` which are not valid Polish expression and do not correspond to any layouts.

Crossovers and Mutations In order to use the Polish expression as chromosomes, [CHMR91] suggested four types of special crossovers CO1 to CO4 and three types of mutations MU1 to MU3 as shown in Table 5.1.

At the crossover stage, one of four types of crossovers is randomly chosen and applied for the parents. For example, if the crossover rate is 0.5, each type of crossover occurs with 0.125 probability. When CO3 is chosen, a sub-tree is randomly selected in one parent to keep the structure of the sub-tree. When CO4 is chosen; a sub-tree is randomly selected in each parent so that both sub-trees can have the same size, and the sub-tree of the first parent is replaced with that of the second parent.

For mutations, [CHMR91] used [WL86]'s methods which were originally used for moves in simulated annealing. The descriptions are also indicated in Table 5.1. At the mutation stage, one of three types of mutations is randomly chosen and applied. So, for example, if the mutation rate is 0.3, each type of mutation occurs with 0.1 probability.

Population Size [CHMR91] used a parallel GA, where there are separately evolving N populations which receive S chromosomes from other populations in every E generations. To keep the population size of each population equal, n out of $(n+S)$ chromosomes are randomly chosen by a fitness based selection method, which will be described below. In the study, they used $n = 80, N = 4, E = 16$, but S was not reported. So, in my implementation, I assumed that $S = 1$.

Selection Method As for selection, [CHMR91] used the following method. First, the mean and the standard deviation of chromosome scores (i.e. layout scores) are calculated in each population. Suppose they are μ and σ. Then, each chromosome's score X is converted to the fitness value F as follows.

$$F = \frac{(\mu - X) + \sigma}{2\sigma}$$

However, if F becomes negative, F will be set as ε, a very small positive number, instead. Since [CHMR91] did not report the value of ε, I assumed it to be 0.01 for my implementation. Finally, the probability of selecting a particular chromosome is proportional to the fitness value F.

Table 5.1. Special crossovers and mutations for the Cea algorithm

crossovers (P1 and P2 mean the parents.)

	STS topology	operators	operands	sub-tree
CO1	same as P1	P2's order	same as P1	N/A
CO2	same as P1	same as P1	P2's order	N/A
CO3	same as P1	same as P1	P2's order excluding inside of the sub-tree	produced in P1
CO4	same as P1 excluding inside of the sub-tree	P1's order excluding inside of the sub-tree	P1's order excluding inside of the sub-tree	produced in P1 and P2 of the same size. P1's are replaced with P2's

e.g. by CO1: P1, `132UL4U`, and
P2, `24U13UL`, create `132UU4L`.

by CO2: P1, `132UL4U`, and
P2, `24U13UL`, create `241UL3U`.

by CO3: P1, $\texttt{1}\underline{\texttt{32U}}\texttt{L4U}$, whose sub-tree is $\underline{\texttt{32U}}$ and
P2, `24U13UL`, create $\texttt{4}\underline{\texttt{32U}}\texttt{L1U}$.

by CO4: P1, $\underline{\texttt{132UL}}\texttt{4U4L}$, whose sub-tree is $\underline{\texttt{132UL}}$ and
P2, $\texttt{42U}\underline{\texttt{34L1L}}\texttt{U}$, whose sub-tree is $\underline{\texttt{34L1L}}$ create
$\underline{\texttt{34L1L}}\texttt{2U4L}$.

(_ denotes the sub-tree.)

mutations

MU1	exchange two operands side by side
MU2	complement a series of operators (operators should be `U` or `L`)
MU3	exchange an operand and an operator side by side (The result of MU3 must be valid Polish expression.)

e.g. by MU1: $\texttt{13}\underline{\texttt{2}}\texttt{UL}\underline{\texttt{4}}\texttt{U}$ may be changed to `134UL2U`.
by MU2: $\texttt{132}\underline{\texttt{UL}}\texttt{4U}$ may be changed to `132LU4U`.
by MU3: $\texttt{13}\underline{\texttt{2U}}\texttt{L4U}$ may be changed to `13U2L4U`.
(_ denotes the genes affected by mutations.)

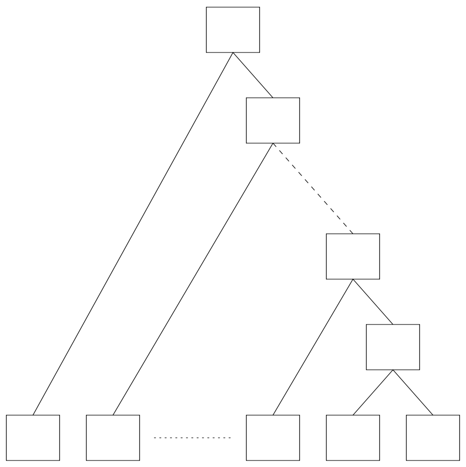


Figure 5.1. The initial STS topology of the Cea algorithm

Initial Chromosomes The chromosome initialisation of the Cea algorithm is as follows. First, every chromosome's STS has the same topology as shown in Figure 5.1. Second, facility indexes are randomly input in terminal nodes so that every index can appear once. Third, either `U` or `R` is chosen for each non-terminal node at random.

5.2 The Tam algorithm

Chromosome Representation Whereas the Cea algorithm used ordinary Polish expressions directly for the chromosome representation, the Tam algorithm used the reduced Polish expressions, which is introduced in Section 4.4, to permit conventional crossovers and mutations. That is, in the Tam algorithm, a chromosome includes only operators of the Polish expression in the same order.

In order to use the reduced Polish expressions, the Tam algorithm first fixes the topology and terminal nodes of the STS by a clustering method, which is called average linkage method. Because this partially fixed STS is used during the search, the reduced Polish expression can represent the physical layouts. For instance, if the partially fixed STS is decided as shown in Figure 5.2(a), the reduced Polish expression, `zywx` can represent the ordinary Polish expression, `12z3y45wx`. In order to decode the reduced Polish expressions, the Polish expression's template such as `12%3%45%%` may be used as described in Section 4.4.

Table 5.2. Average linkage clustering method (for the Tam algorithm)

Step 1:	Pick up a pair of facilities which have the highest traffic.
Step 2:	Regard the pair as a cluster.
Step 3:	Pick up a pair of two facilities (or clusters) which have the highest† traffic.
Step 4:	Go to Step 2 until all the facilities are included in one cluster.

† The traffic between a cluster pair is defined by the average traffic between each facility in a cluster and each facility in the other cluster.

[Tam92a] mentioned that this representation may be effective because the search space becomes much smaller than the ordinary case such as the Cea algorithm. Actually as mentioned in Section 4.4, the search space size of the Tam algorithm is only 2^{2N-2} $(= 4^{N-1})$, while that of the Cea algorithm is at least $2^{3N-4} \cdot N!$. But since the search space is limited, this algorithm can not produce any solutions which can not be expressed by the partially fixed STS.

Average Linkage Clustering To decide the topology and terminal nodes of the STS, [Tam92a] used an average linkage clustering method as shown in Table 5.2.

Here, I will show an example of this method. Suppose that a traffic matrix is given as shown in Table 5.3(a). First, the facilities No.1 and No.2 are picked up because they have the highest traffic link. Although No.4 and No.5 have the traffic link of same frequency, this clustering method arbitrarily picks the pair. Then, the facilities No.1 and No.2 are regarded as one facility A, and the traffic matrix is recalculated as shown in Table 5.3(b). At that time, the calculation will be done based on the average linkage. (e.g. the traffic between A and No.3 is the average of traffic between No.1 and No.3 and that between No.2 and No.3.) After that, because the facilities No.4 and No.5 have the highest traffic, they are picked up and regarded as one facility B. Then, the traffic matrix is recalculated again. (See Table 5.3(c)) Repeating similar operations until all the facilities are merged into one cluster, the topology and terminal nodes of STS are decided.

Table 5.3. An example of traffic matrix

	No.1	No.2	No.3	No.4	No.5
Facility No.1	-	5	2	4	1
No.2	5	-	3	0	2
No.3	2	3	-	0	0
No.4	4	0	0	-	5
No.5	1	2	0	5	-

(a)initial matrix

	A	No.3	No.4	No.5
Facility A	-	2.5	2	1.5
No.3	2.5	-	0	0
No.4	2	0	-	5
No.5	1.5	0	5	-

(b) after merging No.1 and No.2 into A

	A	No.3	B
Facility A	-	2.5	1.75
No.3	2.5	-	0
B	1.75	0	-

(c) after merging No.4 and No.5 into B

In this example, the STS will become as shown in Figure 5.2(a). So, the Polish expression will be `12z3y45wx` where `x`, `y`, `z` and `w` express some operators, and the chromosome is represented as `zywx`.

GA Environments Here, I give details of [Tam92a]'s GA parameters and environments for reference. For initialisation, one of four operators: `U, B, R, L` is set in every gene. And he set population size = 30.

Regarding crossovers and mutations, [Tam92a] used conventional methods. However, the definition of mutation rate is different from my definition in Chapter 2. While I defined that mutation will occur on each allele with the probability of mutation rate, [Tam92a] defined that there is a mutation rate's change that

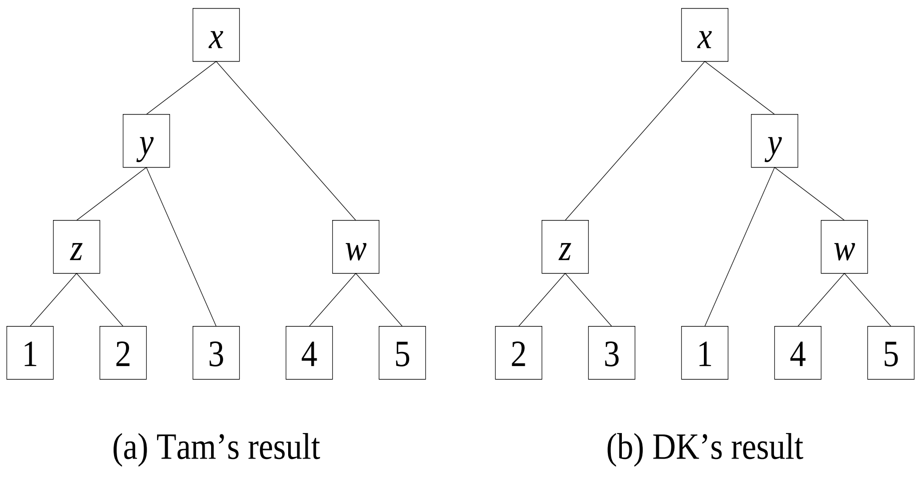


Figure 5.2. Clustering results

a structure in the population will be changed by altering one of its symbols. So, if the length of chromosomes is $M - 1$, and if the mutation rate is small enough; then, the relation between mutation rate of my definition, R_m, and that of [Tam92a], R_t, will be $(M-1)R_m \approx R_t$.

As regards the selection of chromosomes, he used the following method. First, the mean and the best of chromosome's scores are obtained. Suppose they are μ and X_{best}. Then, each chromosome's score X is converted to the fitness value F as follows.

$$F = \frac{\mu - X}{\mu - X_{best}} \times 0.8 + 1$$

However, if F becomes negative, F will be set as ε, a very small positive number, instead. Since [Tam92a] did not report the value of ε, I assumed it to be 0.01 for my implementation. Finally, the probability of selecting a particular chromosome is proportional to the fitness value F.

5.3 The DK algorithm

The DK algorithm is same as the Tam algorithm except that it uses another clustering method, a total linkage clustering. Because this clustering method is introduced by [DK85], I would like to call this algorithm DK here.

Table 5.4. Total linkage clustering method (for the DK algorithm)

Level 1:	Divide all the facilities into two groups† using Steps 1 to 3 below.	
	Step 1:	Separate the dividing facilities into two groups† arbitrarily.
	Step 2:	Consider the possibility of swapping a facility of each group over so that it can reduce the traffic between the groups.
	Step 3:	If it is possible, do it and go to Step 2. Otherwise, go to next level.
Level 2:	Divide the facilities in each group into two sub-groups† using Steps 1 to 3 above.	
	:	
Level N:	Repeat the same operation until each sub-sub-...-group consists of one facility.	

† At that time, the number of facilities in each group should be the same or different by one.

Total Linkage Clustering Whereas [Tam92a]'s clustering method is a sort of bottom-up approach, [DK85]'s is a sort of top-down approach. The process of this clustering method is shown in Table 5.4.

To describe this method, I will show an example using the traffic matrix shown in Table 5.3(a).

In Level 1, the facilities are separated into two-facility group and three-facility group because the number of facilities is five. At Step 1, two groups are arbitrarily created. Suppose that they are facilities No.1 and 2 and facilities No.3, 4 and 5. As Steps 2 and 3, swapping a facility of each group over is considered. For example, facilities No.1 and No.3 may be swapped round because it reduces the total traffic between two groups from 12 to 9 as shown in Figure 5.3. However, after this swapping, no other swapping will occur because there are no facility pair which can reduce the traffic if they are swapped over. Therefore, the clustering of the first level is finished as shown in Figure 5.3(b).

In Level 2, the three-facility group (i.e. facilities No.1, 4 and 5) is similarly

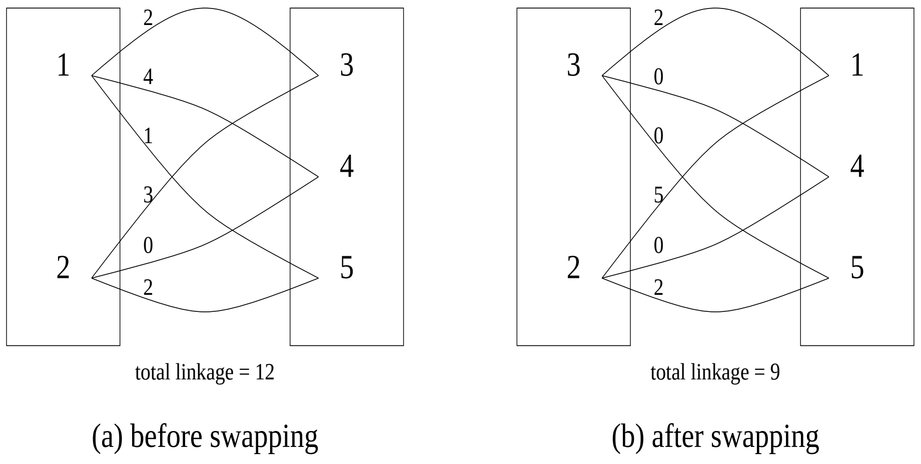


Figure 5.3. An example of swapping facilities over

separated into two sub-groups arbitrarily. As a result, one subgroup may include facility No.1, while the other may include facilities No.4 and 5.

Finally, the result may become as shown in Figure 5.2(b). So, in this case, the clustering result by [DK85]'s method is different from that by [Tam92a]'s method as shown in Figure 5.2, though both of them started from the same traffic matrix shown in Table 5.3(a). Consequently, the search space and the performance of these two algorithms may be different owing to the clustering results.

5.4 Other Possible GAs

Comparing the three GAs mentioned in the previous sections, we may hit upon some ideas for enhancement of these GAs. Here, I will discuss this matter from three points of view: the search space; the chromosome representation; and the clustering method.

The Search Space As already mentioned, the Tam and DK algorithms limit the search space in order to use the reduced Polish expressions. Therefore, if there is a good solution within the limited space, these algorithms may reach it with high probability due to their intensive search in the space. But, if there are no good solutions in the limited space, they may get very poor results. Consequently, how to find out a good potential search space (i.e. a good clustering method) should be an important issue in FLPs. However, finding out a good search space may be highly dependent on FLP's specification; accordingly, I think it is more interesting to consider how to widen the search space from the limited space once specified by a certain clustering method.

The Chromosome Representation Since the Cea algorithm uses Polish expressions as the chromosome representation, it requires special crossovers and mutations as already mentioned. However, the mechanism of the special crossovers and mutations are much more complex than that of conventional ones. Therefore, this may slow the GAs down. So, it may be useful if we can establish a representation method which represents the Polish expression naturally and which permits conventional operations of the crossovers and mutations.

As a possible approach, [WO94]'s method may be mentioned. It is originally introduced for Genetic Programming (GP) rather than GAs. Because GP treats tree structures with varied topology, the idea may be applicable for FLPs. [WO94]'s idea is as follows. First, a basic tree is defined so that it can involve all tree structures which may appear in a particular problem. Second, using a part of the basic tree and *non-branch* operators: `<` and `>`, each chromosome is represented. Third, the unused area of the basic tree is filled by dummy operands or dummy operators at random.

To explain this idea, I will show an example. If we assume that the depth of the basic tree is at most three, the basic tree's topology is decided as shown in Figure 5.4(a). Then, if we want to represent a tree shown in Figure 5.4(b), it may be expressed like Figure 5.4(c), where `<` (or `>`) operator means *"See the left (or right) child node."* Since these operators only refer to one child node, I would like to call them *non-branch* operators. Incidentally, I will call ordinary operators, `U,B,R,L`, *branch* operators.

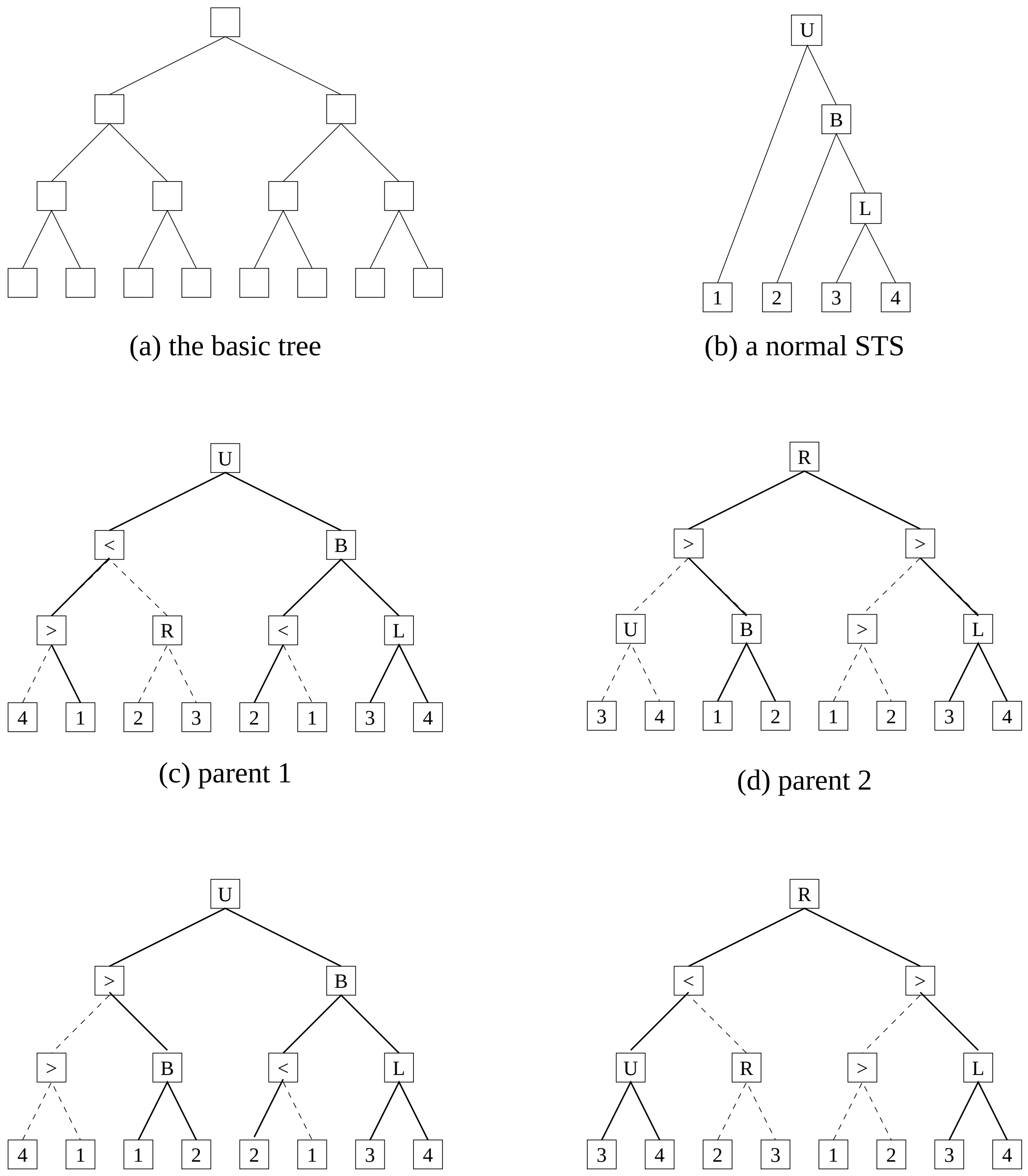
Because the topology of the basic tree is constant, the places of operators and operands are constant in the Polish expressions corresponding to the basic tree. For instance, as for the Polish expression corresponding to the basic tree shown in Figure 5.4(a); the first and second places must be occupied by operands, the third place must be occupied by an operator, and so on. Therefore, if the Polish expressions are used as the chromosome representation, the conventional crossovers and mutations can be applied for this tree. For example, if two chromosomes `41>23R<21<34LBU` and `34U12B>12B34L>R` corresponding to the trees shown in Figure 5.4(c) and (d) are two-point crossed over, two children `41>12B>21<34LBU` and `34U23R<12B34L>R` corresponding to the trees shown in Figure 5.4(e) and (f) may be produced. Of course, they are valid chromosomes in the GP.

However, this [WO94]'s idea is unfortunately difficult to be applied for FLPs immediately. This is because the number of facilities included in a child may be different from the parent's (See Figure 5.4(e)), and because some facility may appear more than once or may not be used at all (See Figure 5.4(f)). That is, the restrictions regarding the number of facilities may be so strong in FLPs that even an idea which is suitable for GPs may be inapplicable directly.

Of course, the idea is not completely impossible. For an FLP involving N facilities, the non-terminal nodes can be filled by integers which index into a circular list consisting of operators, `U,B,R,L,<,>`, rather than the operators themselves; and the terminal nodes can be filled by integers which index into a circular list consisting of facility indexes, $0, 1, \cdots, N$, rather than the facility indexes themselves. Because the number of "active" terminal nodes and "active" non-terminal nodes should be N and $N-1$, respectively; some method which interprets unnecessary branch operators as non-branch operators (and vice versa) may be required.

The Clustering Method Although two types of clustering methods are mentioned in the previous sections, there have been many other clustering methods as shown in [Chu89]. So, using one of them instead of the clustering method of Tam (or DK) algorithm, we can produce many sorts of GAs for FLPs. However, I would like to discuss the possibility of the stochastic clustering technique here rather than using one of the conventional clustering methods.

For instance, suppose that we are clustering facilities by Tam's method. There, we have to pick up two facilities/clusters which have the highest traffic. However, in a particular stage, there might be some alternatives which have the same traffic. For example, if we have the traffic matrix shown in Table 5.3(a); then, either facilities No.1 and No.2 or facilities No.4 and No.5 can be picked up



(a) the basic tree

(b) a normal STS

(c) parent 1

(d) parent 2

(e) child 1

(f) child 2

Figure 5.4. [WO-94]'s representation because they have the same traffic. Though Tam's method decides the facility/cluster pair deterministically, it could choose at random in such cases.

Moreover, even if we do not meet such cases, we could use stochastic approach for picking up facility/cluster pair. That is, a pair which does not have the highest traffic might be picked up with less probability than the highest pair.

In summary, by using such a stochastic approach, many clustering results will be produced, and they may be able to scatter initial chromosomes over the search space. At that time, if we use reasonable clustering methods, the initial chromosomes may locate in the neighbourhood of excellent solutions and this might make the search quite effective.

5.5 The Tam2, DK2 and Kad algorithms

Among the basic ideas above, I have tried to expand the search space of the Tam and DK algorithms. Here, I will describe my approach.

Chromosome Representation In order to expand the search space of the Tam algorithm, I modified the chromosome representation as follows.

reduced Polish expression (e.g. `RLU`)	Polish expression's template (e.g. `12%34%%`)

The first part of the chromosome is the reduced Polish expression which is same as Tam's (or DK's) representation. The second part specifying the topology and terminal nodes of the STS is the Polish expression's template, which is introduced in Section 4.4. For example, the STS shown in Figure 5.5(a) corresponding to the Polish expression `12R34LU` is encoded as `RLU12%34%%`. This is because its reduced Polish expression is `RLU` and because its template is `12%34%%`, where `%` means an operator's position.

Crossovers and Mutations For this representation, if the second part is regarded as one gene, the similar operations to conventional crossovers and mutations can be applied. For instance, because the splitting point of crossovers will

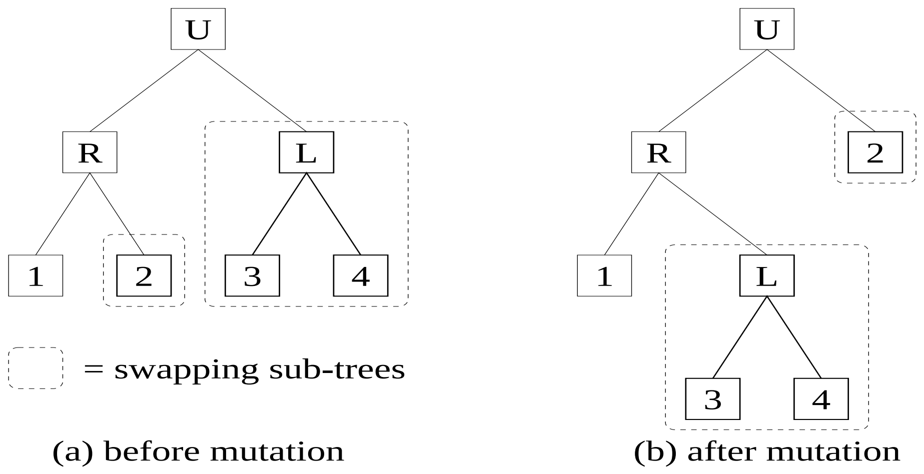


Figure 5.5. A mutation of the Tam2/DK2/Kad algorithms

not be set inside of the second part, crossovers will not break the STS topology. That is, the child created by crossovers is always valid representation of a layout.

In the mutation, if the second part is chosen to be mutated, the STS topology will be changed by swapping two subtrees over. At that time, the subtrees are randomly chosen so that they can not overlap each other. For instance, if an STS and subtrees are given as shown in Figure 5.5(a), the mutated STS will be as shown in Figure 5.5(b) and the chromosome will be `LRU134%%2%`.

The Tam2 and DK2 algorithms Applying the above representation and operations of crossovers and mutations to the Tam and DK algorithms, new GAs which can search larger space than the original algorithms can be obtained. This is because if a mutation happens on the Polish expression's template, the new GAs can change the STS topology and/or the terminal nodes of the STS. For convenience, I will call the new GAs the Tam2 and DK2 algorithms, respectively. All the other specifications of new GAs are same as original GAs.

The Kad algorithm When we use the above representation and operations of crossovers and mutations, we do not have to set up all the chromosomes so that they can have the same STS topology. Therefore, instead of using one clustering result, I tried to use both results of Tam's and DK's clustering methods. Here, 50% of initial chromosomes have Tam's clustering result and the other 50% have

DK's clustering result. For convenience, I will call this algorithm Kad. Similarly, the Kad algorithm uses the same GA parameters, environments, etc. of the Tam and DK algorithms.

5.6 Summary

As a summary, I put the specifications of six GAs in Table 5.5.

Table 5.5. The specifications of six GAs

name	representation	crossovers and mutations	the topology and operand positions of the initial STS	operators	search space (STS topology)
Cea	Polish expression	special operations (See Table 5.1)	fixed topology (See Table 5.1) and random operands	`U` and `L` only	free
Tam	reduced Polish expression	ordinary operations	decided by the average linkage method (See Table 5.2)	`U,B,L,R`	limited in the initial topology
DK	same as Tam	same as Tam	decided by the total linkage method (See Table 5.4)	`U,B,L,R`	limited in the initial topology
Tam2	reduced Polish expression + its template	special operations (See Section 5.5)	same as Tam	`U,B,L,R`	free
DK2	same as Tam2	same as Tam2	same as DK	`U,B,L,R`	free
Kad	same as Tam2	same as Tam2	50% are same as Tam and the others are same as DK	`U,B,L,R`	free

Chapter 6

Experiments and Results

6.1 The Design of Experiments

6.1.1 The Objectives of Experiments

As already mentioned in Section 2.3, my research interests are:

- Investigating GA parameters to find out if there are some special combinations which are effective to FLPs independent of specific problems
- Comparing GA performance with other algorithms based on the standard problems
- Comparing different GAs performance with each other from the perspective of STS usage

I first did experiments about the investigation of GA parameters. Then, using the results of those experiments, I compared six types of GAs with each other as well as with other algorithms (SA and QN). In this chapter, I will describe these experiments and discuss the results, after introducing some tools of my experiments.

6.1.2 PGA Program

PGA (Parallel GA) is a GA simulator which has been developed in Edinburgh University AI department [RH95]. The original program was designed by Ballinger,

Table 6.1. PGA usage for FLPs

`lop-pga`	[`-r`*algorithm*] [`-e`*problem_file*] [`-s`*selection_method*]
	[`-P`*number_of_populations*] [`-p`*population_size*]
	[`-m`*mutation_rate*] [`-c`*crossover_rate*]
	[`-C`*crossover_points*] [`-t`]
algorithm =	`lop`*rrraaa*
	where *rrr* = reproduction-method
	(`gen`: generation-based)
	(`one`: Genitor)
	and *aaa* = GA's name
	(`Cea, Tam, DK, Tam2, DK2, Kad`)
problem_file =	`lop`*file*
	where *file* = the name of FLP specification file (e.g. `Kea91-11`)
selection_method =	`rank`, `tn`*S* or `tm`*S*
	where *S* = tournament size
	(`tn` = tournament selection)
	(`tm` = modified tournament selection)
crossover_points =	`one` or `two`
	showing the number of crossover points
`-t` =	If added, two complementary children will be
	produced per generation. This option is valid
	only for Genitor reproduction.

but the current version is due to Ross, who has designed it partly to serve as a starting point for various GA applications. Based on this original program, I added some functions to solve FLPs. Table 6.1 shows a brief usage of PGA related to FLPs.

6.1.3 Statistical Tests

To compare the performance of two or more algorithms, I used three types of statistical tests: t-tests, F-tests and protected t-tests.

Table 6.2. The procedure of a t-test

Step 1:	obtain each sample's mean m_i, standard deviation s_i, and sample size n_i; where $i = 1$ or 2.
Step 2:	estimate the combined standard deviation of both samples s_{com} $s_{com} = \sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2} \times (\frac{1}{n_1} + \frac{1}{n_2})}$
Step 3:	calculate t-value t and degree of freedom df $t = \frac{m_1-m_2}{s_{com}}$, $df = n_1 + n_2 - 2$
Step 4:	obtain the critical t-value t_{crit} on df and on the criterion of significance.
Step 5:	if $t < -t_{crit}$ or $t_{crit} < t$, then the sample's difference can be said to be significant.

T-tests T-tests are sometimes used for testing if there is a substantial difference between the means of two sampled populations [WEC91].

The t-tests require the following assumptions. First, both of the two samples follow the normal curve model. Second, the two samples have the same standard deviations. Therefore, if the sizes of two samples are significantly different, t-tests may not be useful for evaluations. In my thesis research, I will use t-tests to compare two sets of samples only if both sets contain at least ten samples. The procedure of a t-test is shown in Table 6.2. Using t-tests, we can evaluate whether some algorithm's performance are probably significantly different or not.

F-tests Whereas the t-test is useful for comparing two samples, it is not suitable for comparing more than two samples because many t-tests may increase statistical error. [WEC91].

Suppose that the means of samples to be compared are $m_1, m_2, \ldots m_k$ where k is the number of samples. Then, $k(k-1)/2$ t-tests are necessary to compare each pair of the samples (i.e. m_i and $m_j (i \neq j)$). However, each t-test has some criterion of significance; therefore, many t-tests may cause substantial statistical error. For instance, if the criterion of significance is 5% and if k is 5, then the expected value of the number of errors is 0.5 ($0.05 \times 5(5\text{-}1)/2 = 0.5$).

So, instead of the t-tests, F-tests are sometimes used in such cases. In F-test, F-ratio which is the ratio of the within-group variation to the between-group variation, is used for the decision if m_i are all same. That is, the F-ratio should be large when the means of samples are significantly different. The procedure of F-test is shown in Table 6.3 [WEC91].

Protected t-tests When F-test shows that the means of groups are different, protected t-tests are sometimes used to know which sample's mean is significantly different from another. Unlike ordinary t-tests, the protected t-tests calculate t-scores by combining standard deviation of all groups as follows. [WEC91]

$$t = \frac{m_1 - m_2}{\sqrt{MS_W(\frac{1}{N_i} + \frac{1}{N_j})}}$$

At that, $df = \sum_i^k N_i - k$ is used to define the degree of freedom.

6.2 GA Parameters Investigation

6.2.1 The Experiments

In addition to the GA parameters investigated by [SCED89], three types of selection methods are added for my investigation. Table 6.4 shows the GA parameters which I used in my investigation.

The table includes $12 \times 7 \times 6 \times 2 \times 3 = 3024$ variations of GA parameters. Therefore, if we investigate each combination ten times, 453600 experiments are required for fifteen FLPs. (i.e. $453600 = 3024 \times 10 \times 15$) However, this number is so large that I used a hill-climbing investigation style rather than a full investigation style. The procedure of hill-climbing investigation is shown in Table 6.5

In this hill-climbing investigation, only 30 (= 12+7+6+2+3) variations will be investigated. Because this number is about 1% of the full investigation, this hill-climbing investigation ignores most of the space of GA parameters variation. Therefore, the final set of parameters chosen is potentially a local optimum in

Table 6.3. The procedure of a F-test

Step 1:	calculate total sum of squares (SS_T)
	$SS_T = \sum X^2 - \frac{(\sum X)^2}{N}$
	where X = score of each observation
Step 2:	calculate sum of squares between groups (SS_B)
	$SS_B = \frac{(\sum X_1)^2}{N_1} + \frac{(\sum X_2)^2}{N_2} + \cdots \frac{(\sum X_k)^2}{N_k} - \frac{(\sum X)^2}{N}$
	where
	$\sum X_i$ = sum of scores in group i
	k = number of groups
	N_i = number of scores in group i
Step 3:	obtain sum of squares within groups (SS_W)
	$SS_W = SS_T - SS_B$
Step 4:	obtain the degrees of freedom between groups (df_B) and within groups (df_W)
	$df_B = k - 1$
	$df_W = N - k$
Step 5:	calculate F-score as the ratio of the mean squares between groups
	(symbolised by MS_B) and within groups (symbolised by MS_W)
	$F = \frac{MS_B}{MS_W} = \frac{SS_B/df_B}{SS_W/df_W}$
Step 6:	obtain the critical F-score F_{crit} on df_B and df_W and
	on the criterion of significance
Step 7:	if $F < -F_{crit}$ or $F_{crit} < F$, the means of groups can be
	said to be different.

Table 6.4. GA parameters to be investigated

investigating parameters	
crossover rate	0, 0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95, 1
mutation rate†	0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1
population size	10, 20, 30, 50, 100, 200
crossover points	one point, two points
selection methods	rank, tournament, modified tournament (tournament size = 5)
common parameters	
reproduction method	generation-based
number of populations	1
max number of generations	100

†In the Tam and DK algorithms, the conventional mutation will occur on each allele with the probability of mutation rate. In the Cea algorithm, one of three types of mutations shown in Table 5.1 is applied with the probability.

Table 6.5. The procedure of hill-climbing investigation

Step 1:	*default parameters set up* A combination of GA parameters to be investigated is selected arbitrarily. Call them default parameters.
Step 2:	*crossover rate investigation* To find the best crossover rate among twelve alternatives shown in Table 6.4, do ten experiments for each alternative. At that time, set the other GA parameters as the default. Choose the best crossover rate among the alternatives by seeing the best individual performance, and regard it as the new default parameter of crossover rate.
Step 3:	*mutation rate investigation* Decide the best mutation rate from seven alternatives in similar way to Step 2, and regard it as the new default parameter of mutation rate.
Step 4:	*crossover points investigation* Decide the best number of crossover points from two alternatives in similar way to Step 2, and regard it as the new default parameter of the number of crossover points.
Step 5:	*selection method investigation* Decide the best selection method from three alternatives in similar way to Step 2, and regard it as the new default parameter of the selection method.
Step 6:	*population size investigation* Decide the best population size from six alternatives in similar way to Step 2, and regard it as the new default parameter of the population size.
Step 7:	*final results* Regard the final set of default parameters as the best GA parameters.

Table 6.6. The best GA parameters (the Cea algorithm)

problem	population size	selection method	crossover rate	mutation rate
Kea91-11	200	tn5	0.95	0.1
Kea91-11a	200	tm5	1.00	0.1
Kea91-16	200	rank	0.75	0.005
Kea91-20	200	tn5	0.95	0.05
TL91-5	200	tm5	0.25	0.1
TL91-6	200	rank	0.65	0.1
TL91-7	200	rank	0.35	0.1
TL91-8	200	tn5	0.85	0.1
TL91-12	200	rank	0.95	0.1
TL91-15	200	tn5	0.55	0.1
TL91-20	200	tm5	0.75	0.1
TL91-30	200	rank	0.75	0.05
Tam92-20a	200	tm5	0.45	0.001
Tam92-30a	200	tm5	0.95	0.001
VCea91-10	200	rank	0.05	0.1

Because the Cea algorithm uses special crossover methods, the investigation for the number of crossover points was not done.

this space. Nevertheless, I used hill-climbing method to save time. If precise results are necessary, further iterations of Steps 2 to 6 in Table 6.5 could be done.

6.2.2 Results and Discussions

Solving the fifteen standard FLPs using the Cea, Tam and DK algorithms, I obtained the best GA parameters shown in Tables 6.6 to 6.8, which are summarised in Table 6.9. In the table, the number in each slot stands for how many times the given parameter led to the best solution using the given algorithm.

From Table 6.9, we can conclude as follows. As for crossover rates and the number of crossover points, no particular value shows outstanding performance. Regarding mutation rates, the value around 0.01 may be most suitable for the Tam/DK algorithms, whereas the value of 0.1 seems more suitable for the Cea

Table 6.7. The best GA parameters (the Tam algorithm)

problem	population size	crossover points	selection method	crossover rate	mutation rate
Kea91-11	200	two	tm5	0.55	0.020
Kea91-11a	200	two	tn5	0.25	0.010
Kea91-16	200	two	tm5	0.25	0.005
Kea91-20	200	two	rank	0.00	0.010
TL91-5	20	two	tn5	0.25	0.010
TL91-6	100	one	tn5	0.15	0.010
TL91-7	200	one	tn5	0.55	0.010
TL91-8	50	two	rank	0.45	0.010
TL91-12	200	one	tn5	0.85	0.010
TL91-15	50	one	rank	0.75	0.010
TL91-20	200	one	tm5	1.00	0.005
TL91-30	200	two	tm5	0.55	0.005
Tam92-20a	100	two	tm5	1.00	0.001
Tam92-30a	200	two	tm5	0.85	0.005
VCea91-10	200	one	tm5	1.00	0.010

Table 6.8. The best GA parameters (the DK algorithm)

problem	population size	crossover points	selection method	crossover rate	mutation rate
Kea91-11	200	one	tn5	0.95	0.01
Kea91-11a	200	one	tn5	0.95	0.01
Kea91-16	50	two	tm5	0.00	0.01
Kea91-20	200	two	rank	0.45	0.01
TL91-5	100	two	tm5	0.25	0.005
TL91-6	50	one	tn5	0.65	0.02
TL91-7	20	two	tn5	0.95	0.05
TL91-8	50	two	rank	0.15	0.01
TL91-12	200	one	tm5	0.05	0.02
TL91-15	100	one	tn5	0.25	0.02
TL91-20	200	one	tn5	0.85	0.01
TL91-30	200	two	tn5	0.75	0.002
Tam92-20a	30	one	tm5	1.00	0.005
Tam92-30a	200	two	tm5	0.65	0.01
VCea91-10	200	two	tn5	0.25	0.02

Table 6.9. A summary of GA parameters investigation

Crossover Rate

	0.00	0.05	0.15	0.25	0.35	0.45	0.55	0.65	0.75	0.85	0.95	1.00
Cea		1		1	1	1	1	1	3	1	4	1
Tam	1		1	3		1	3		1	2		3
DK	1	1	1	3		1		2	1	1	3	1
total	2	2	2	7	1	3	4	3	5	4	7	5

Mutation Rate

	0.001	0.002	0.005	0.010	0.020	0.050	0.100
Cea	2		1			2	10
Tam	1		4	9	1		
DK		1	2	8	4		
total	3	1	7	17	5	2	10

Population Size

	10	20	30	50	100	200
Cea						15
Tam		1		2	2	10
DK		1	1	3	2	8
total		2	1	5	4	33

Crossover Points and Selection Methods

	one	two	rank	tn5	tm5
Cea	-	-	6	4	5
Tam	6	9	3	5	7
DK	7	8	2	8	5
total	13	17	11	17	17

The number in each slot stands for how many times the given parameter led to the best solution.

algorithm. In the selection method investigation, the Tam/DK algorithms may obtain best results with the (modified) tournament selection methods, while the Cea algorithm reaches good results with rank method. As regards the population size, larger populations are best in every algorithm, especially Cea.

Although most of these findings can not be explained clearly, I can say the following.

First, higher mutation rate is preferable in the Cea algorithm than in the Tam and DK algorithms. This is probably because the definition of mutation rate for Cea is different from that for Tam/DK. That is, while the mutation rate for Tam/DK is the probability of mutating each allele of a chromosome, that for Cea is the probability of mutating a chromosome. For example, if we assume M is the mutation rate and L is the length of the chromosome (i.e. the number of alleles), the probability of mutating a chromosome in the Cea algorithm is M and that in Tam/DK is $L \times M$ approximately. In addition, since Cea generally searches a larger space than Tam/DK as already mentioned in Chapter 4, changing STS topology may be critical for this algorithm. Because the Cea algorithm sets up initial chromosomes so that they have the same STS topology as shown in Fig 5.1, they cannot change their STS topology without MU3 operations (Table 5.1). So, in order to escape from the initial mediocre STS topology, the Cea algorithm might need higher mutation rate than the Tam/DK algorithms. Actually, I did another mutation rate investigation for the Cea algorithm using the mutation rate more than 0.1 as shown in Table 6.10. As the result of the investigation, I confirmed higher mutation rate than 0.1 certainly showed better results in Cea as shown in Table 6.11.

Second, the good performance of larger population sizes is probably due to the larger number of evaluations. Because the experiments were done in the same number of generations (i.e. constantly 100), the GA with larger population size produced and evaluated more chromosomes. Considering the fact that this tendency is stronger in the Cea algorithm, the Cea algorithm may need much more evaluations than the Tam/DK algorithms to discover solutions in its larger search space. However, apart from the number of evaluations, the larger population size itself might have some effects in FLPs. So, this issue will be investigated in the experiment in the next section.

Table 6.10. The specification of additional mutation rate investigation for the Cea algorithm

investigating parameters	
mutation rate	0.1, 0.2, 0.4, 0.6, 0.8, 1.0
other specifications	
algorithm	Cea
problems	the fifteen FLPs
population size	the best parameter shown in Table 6.6
selection method	*same as above*
crossover rate	*same as above*
reproduction method	generation-based
number of populations	1
max number of generations	100

Table 6.11. The result of additional mutation rate investigation for the Cea algorithm

problem	the best mutation rate	problem	the best mutation rate
Kea91-11	0.2	TL91-12	0.8
Kea91-11a	0.6	TL91-15	1.0
Kea91-16	1.0	TL91-20	1.0
Kea91-20	0.8	TL91-30	1.0
TL91-5	1.0	Tam92-20a	1.0
TL91-6	0.2	Tam92-30a	1.0
TL91-7	0.6	VCea91-10	0.4
TL91-8	1.0		

Summary

mutation rate	0.1	0.2	0.4	0.6	0.8	1.0
*frequency		2	1	2	2	8

(*) The frequency stands for how many times the given mutation rate led to the best solution.

6.3 Algorithm Comparison

6.3.1 The Experiments

In Chapter 4, six GAs (Cea, Tam, DK, Tam2, DK2, Kad) for FLPs were introduced. Here, I will compare these algorithms' performance with each other and with other algorithms shown in Figure 3.1. In addition, I will investigate the effects of population size, the number of populations and reproduction method as well. The specifications of these experiments are shown in Tables 6.12 to 6.16.

As for mutation rate, the higher probability of 0.8 is set for the Cea algorithms whereas the probability of 0.02 is set for the other five algorithms. This is because the definition of the mutation rate for Cea is different from the others and because much higher mutation rate may be preferable in the algorithm as already mentioned in the previous section.

Similarly, higher value of 1000 is mainly used in this comparison. It is because the observation that higher population size seems to be preferable was found in the previous investigation.

As regards other GA parameters (i.e. crossover rates, the number of crossover points and selection methods), traditional value or methods are chosen as default. However, the best parameters' combination found in the previous experiments are also used for GAs comparison with other algorithms.

Table 6.12. The specification of GAs comparison

algorithms:	**Cea, Tam, DK, Tam2, DK2, Kad**
problems:	15 FLPs
crossover rate:	0.65
crossover points:	two
mutation rate:	0.8 (for the Cea algorithm)
	0.02 (for the other algorithms)
selection method:	rank
population size:	1000
reproduction method:	generation based (*gen*) and
	Genitor with twin children (*two*)
max number of generations:	200 (for *gen*), 100000 (for *two*)
	(i.e. max number of evaluations = 200000)
number of populations:	1
experiments:	10 times for each variation

Table 6.13. The specification of GAs comparison with other algorithms

algorithms:	**Cea, Tam, DK, Tam2, DK2, Kad**
	against **simulated annealing** ([KJK91], [Tam92b])
	and **quasi-Newton methods** ([TL91], [VCCV91])
problems:	15 FLPs
crossover rate:	0.65 or *best*
crossover points:	two or *best*
mutation rate:	0.8 (for the Cea algorithm),
	0.02 (for the other algorithms) or *best*
selection method:	rank or *best*
population size:	50, 200, 1000 or *best*
reproduction method:	generation based (gen)
	Genitor with single child (one) or
	Genitor with twin children (two)
number of populations:	1, 4, 10
max number of evaluations:	as with non-GA algorithms
comparison method:	t-test (if original paper contains many samples)
	comparison of the best results (otherwise)

Best stands for the best GA parameters shown in Tables 6.6, 6.7, 6.8 and 6.11.

Table 6.14. The specification of population size investigation

population size:	**50, 200, 1000**
algorithms:	Cea, Tam, Kad
problems:	15 FLPs
crossover rate:	0.65
crossover points:	two
mutation rate:	0.8 (for the Cea algorithm)
	0.02 (for the other algorithms)
selection method:	rank
reproduction method:	generation based (gen)
the number of generations:	4000, 1000, 200
	(so that max number of evaluations = 200000)
the number of populations:	1
experiments:	10 times for each variation

Table 6.15. The specification of reproduction methods investigation

reproduction method:	**generation based** (*gen*)
	Genitor with single child (*one*) and
	Genitor with twin children (*two*)
algorithms:	Cea, Tam, Kad
problems:	15 FLPs
crossover rate:	0.65
crossover points:	two
mutation rate:	0.8 (for the Cea algorithm)
	0.02 (for the other algorithms)
selection method:	rank
population size:	1000
the number of generations:	200 (for *gen*), 200000 (for *one*), 100000 (for *two*)
	(i.e. max number of evaluations = 200000)
the number of populations:	1
experiments:	10 times for each variation

Table 6.16. The specification of the investigation for the number of populations

the number of populations:	**1, 4, 10**
algorithms:	Cea, Tam, Kad
problems:	15 FLPs
crossover rate:	0.65
crossover points:	two
mutation rate:	0.8 (for the Cea algorithm)
	0.02 (for the other algorithms)
selection method:	rank
population size:	1000, 250, 100 (so that total population = 1000)
reproduction method:	generation based (gen)
the number of generations:	200 (i.e. max number of evaluations = 200000)
experiments:	10 times for each variation
migration methods:	A chromosome is chosen from one population (the first population is chosen for the first migration, the second for the second and so on) and copied into all the other populations.
migration interval:	10 generations

6.3.2 Results and Discussion

All experimental results are shown in Figures 6.1 to 6.8. In these figures, each number indicates the mean of ten best individual scores after 200000 evaluations, and the names of GAs are indicated by the following form.

$rrraaa$`/P`ggg`.P`ppp

where $rrr =$	reproduction method (`gen, one, two`)
$aaa =$	algorithm (`Cea, Tam, DK, Tam2, DK2, Kad`)
$ggg =$	the number of populations (`1, 4, 10`)
$ppp =$	population size $\times$ the number of populations (`50, 200, 1000`) or `best` (where GA parameters shown in tables 6.6, 6.7, 6.8, and 6.11 are used.)

To compare particular GA pairs, t-tests with the criterion of significance of 0.05 were used. In Figures 6.9 to 6.13 and 6.17 to 6.20, a "better" is shown in the better algorithm's column, and two "-"s are shown in both columns in case no significant difference between the two GAs was found. The detailed results also can be seen in Appendix B.

```
                genCea/      genTam/      genDK/       genTam2/     genDK2/      genKad/
                P1.Pp1000    P1.Pp1000    P1.Pp1000    P1.Pp1000    P1.Pp1000    P1.Pp1000
-------------------------------------------------------------------------------------------
      TL91-5      228.15       228.15       267.97       228.15       235.43       228.15
      TL91-6      361.45       377.84       377.84       361.45       363.09       361.45
      TL91-7      595.89       777.57       690.41       646.22       635.28       656.83
      TL91-8      883.76       963.95      1186.52       936.02       954.58       934.56
  VCea91-10     24440.66     22074.93     23107.02     21622.61     21506.32     21538.61
   Kea91-11      2947.11      3328.10      3241.33      2946.64      3094.82      3004.08
  Kea91-11a      2266.87      2460.45      2758.39      2307.81      2328.79      2308.23
     TL91-12     3614.52      3894.58      4043.69      3749.45      3784.31      3789.74
     TL91-15     8924.41      8400.31      8946.64      8279.95      8632.99      8629.88
   Kea91-16       114.09        68.80        76.19        64.00        66.40        67.80
   Kea91-20       189.89       171.00       172.39       168.69       171.80       168.50
     TL91-20    21266.95     17949.37     18910.92     17651.33     18383.84     17495.76
  Tam92-20a     25069.13     21532.31     21451.31     21128.86     21100.78     21083.19
     TL91-30   101773.54     55368.81     49161.25     55610.78     50345.43     50300.66
  Tam92-30a     52672.19     49162.65     45866.41     45884.94     45802.42     45853.37
-------------------------------------------------------------------------------------------


                twoCea/      twoTam/      twoDK/       twoTam2/     twoDK2/      twoKad/
                P1.Pp1000    P1.Pp1000    P1.Pp1000    P1.Pp1000    P1.Pp1000    P1.Pp1000
-------------------------------------------------------------------------------------------
      TL91-5      228.15       228.15       267.97       228.15       263.98       228.15
      TL91-6      365.38       377.84       377.84       376.20       376.20       372.92
      TL91-7      621.79       777.57       690.41       721.91       687.27       670.17
      TL91-8      892.93       963.95      1186.52       963.95       970.48       963.95
  VCea91-10     22902.32     22025.93     23034.72     22002.13     22094.81     21933.63
   Kea91-11      2945.52      3327.91      3241.33      3005.64      3176.69      3127.93
  Kea91-11a      2329.21      2460.45      2758.39      2342.95      2492.93      2318.58
     TL91-12     4267.77      3873.53      4043.69      3834.51      3779.60      3783.61
     TL91-15    10609.23      8313.61      8910.39      8276.51      8568.38      8267.09
   Kea91-16       135.69        64.00        64.00        64.80        66.00        65.59
   Kea91-20       209.00       167.60       165.10       166.50       165.19       166.50
     TL91-20    28152.28     17464.25     18756.57     17088.95     17771.48     17048.46
  Tam92-20a     25124.67     21303.11     21286.97     21011.15     21152.66     20894.95
     TL91-30   126090.65     53396.93     47944.67     47898.02     46025.80     45472.43
  Tam92-30a     53851.98     48432.75     45370.86     45101.39     44629.99     44436.23
-------------------------------------------------------------------------------------------
```

Figure 6.1. The mean of the best scores in algorithm comparison (after 200000 evaluations)

	genCea/ P1.Pp1000	genTam/ P1.Pp1000	genDK/ P1.Pp1000	genTam2/ P1.Pp1000	genDK2/ P1.Pp1000	genKad/ P1.Pp1000
TL91-5	0.00%	0.00%	17.45%	0.00%	3.19%	0.00%
TL91-6	0.00%	4.53%	4.53%	0.00%	0.45%	0.00%
TL91-7	0.00%	30.49%	15.86%	8.45%	6.61%	10.23%
TL91-8	0.00%	9.07%	34.26%	5.91%	8.01%	5.75%
VCea91-10	13.64%	2.64%	7.44%	0.54%	0.00%	0.15%
Kea91-11	0.05%	12.99%	10.04%	0.04%	5.07%	1.99%
Kea91-11a	0.00%	8.54%	21.68%	1.81%	2.73%	1.82%
TL91-12	0.00%	7.75%	11.87%	3.73%	4.70%	4.85%
TL91-15	7.95%	1.61%	8.22%	0.16%	4.43%	4.39%
Kea91-16	78.27%	7.50%	19.05%	0.00%	3.75%	5.94%
Kea91-20	15.02%	3.57%	4.42%	2.17%	4.06%	2.06%
TL91-20	24.74%	5.28%	10.92%	3.54%	7.83%	2.62%
Tam92-20a	19.98%	3.05%	2.66%	1.12%	0.99%	0.90%
TL91-30	123.81%	21.76%	8.11%	22.30%	10.72%	10.62%
Tam92-30a	18.53%	10.64%	3.22%	3.26%	3.07%	3.19%

	twoCea/ P1.Pp1000	twoTam/ P1.Pp1000	twoDK/ P1.Pp1000	twoTam2/ P1.Pp1000	twoDK2/ P1.Pp1000	twoKad/ P1.Pp1000
TL91-5	0.00%	0.00%	17.45%	0.00%	15.70%	0.00%
TL91-6	1.09%	4.53%	4.53%	4.08%	4.08%	3.17%
TL91-7	4.35%	30.49%	15.86%	21.15%	15.34%	12.47%
TL91-8	1.04%	9.07%	34.26%	9.07%	9.81%	9.07%
VCea91-10	6.49%	2.42%	7.11%	2.31%	2.74%	1.99%
Kea91-11	0.00%	12.98%	10.04%	2.04%	7.85%	6.19%
Kea91-11a	2.75%	8.54%	21.68%	3.36%	9.97%	2.28%
TL91-12	18.07%	7.17%	11.87%	6.09%	4.57%	4.68%
TL91-15	28.33%	0.56%	7.78%	0.11%	3.64%	0.00%
Kea91-16	112.02%	0.00%	0.00%	1.25%	3.12%	2.48%
Kea91-20	26.59%	1.51%	0.00%	0.85%	0.05%	0.85%
TL91-20	65.13%	2.44%	10.02%	0.24%	4.24%	0.00%
Tam92-20a	20.24%	1.95%	1.88%	0.56%	1.23%	0.00%
TL91-30	177.29%	17.43%	5.44%	5.33%	1.22%	0.00%
Tam92-30a	21.19%	8.99%	2.10%	1.50%	0.44%	0.00%

Figure 6.2. Percentage by which each result is worse than the best for a particular problem (corresponding to Figure 6.1)

	genCea/ P1.Pp50	genCea/ P1.Pp200	genCea/ P1.Pp1000	genTam/ P1.Pp50	genTam/ P1.Pp200	genTam/ P1.Pp1000	genKad/ P1.Pp50	genKad/ P1.Pp200	genKad/ P1.Pp1000
TL91-5	228.15	228.15	228.15	228.15	228.15	228.15	228.15	228.15	228.15
TL91-6	361.45	361.45	361.45	377.84	377.84	377.84	371.28	369.65	361.45
TL91-7	613.89	603.86	595.89	777.57	777.57	777.57	685.62	690.41	656.83
TL91-8	911.75	891.77	883.76	995.92	973.09	963.95	947.78	937.00	934.56
VCea91-10	26614.31	24774.32	24440.66	22193.76	22181.24	22074.93	21843.05	21550.28	21538.61
Kea91-11	3057.90	3011.88	2947.11	3344.48	3342.45	3328.10	3064.78	3042.96	3004.08
Kea91-11a	2251.18	2243.71	2266.87	2470.47	2471.65	2460.45	2299.94	2307.56	2308.23
TL91-12	3860.45	3643.20	3614.52	4061.78	3951.69	3894.58	3844.24	3773.40	3789.74
TL91-15	9516.45	9279.54	8924.41	8644.35	8756.32	8400.31	8180.13	8426.51	8629.88
Kea91-16	105.69	102.50	114.09	70.40	69.00	68.80	65.19	68.40	67.80
Kea91-20	195.50	196.69	189.89	168.69	170.39	171.00	168.00	168.00	168.50
TL91-20	19466.58	19100.39	21266.95	17710.20	18011.20	17949.37	17284.03	17107.13	17495.76
Tam92-20a	22502.69	24175.53	25069.13	21514.41	21434.70	21532.31	21219.39	21349.92	21083.19
TL91-30	58339.42	59881.24	101773.54	55878.96	54348.69	55368.81	52654.90	49061.87	50300.66
Tam92-30a	50463.72	49051.12	52672.19	49467.97	49312.13	49162.65	46407.08	45434.55	45853.37

Figure 6.3. The mean of the best scores in population size investigation (after 200000 evaluations)

Comparison of GAs The comparisons of six GAs are shown in Figures 6.1 and 6.2. And some t-test results are shown in Figures 6.9 to 6.13.

Comparing the Cea and the Tam/DK algorithms, a clear characteristic can be seen. As shown in Figure 6.9, Cea beat Tam/DK in most FLPs having a small number of facilities, while Tam/DK completely outperformed Cea in FLPs consisting of large numbers of facilities. Although the border line between smaller and larger FLPs is vague, this feature can be obviously seen in the results of t-tests. This difference may be caused by the size of the search space in each algorithm. Because Cea searches larger space, Cea may be slower to reach good solutions in larger FLPs.

Of course, as fifteen t-tests are used for each comparison, one or possibly two t-test results may be wrong due to the criterion of significance of 0.05. However, the tendency that Cea is strong for small FLPs and that Tam/DK is preferable for large FLPs may be too strong to be rejected due to it.

On the other hand, as shown in Figure 6.10, the number of facilities seems to have no effect on the performance difference between Tam and DK and between Tam2 and DK2. Since Tam/Tam2 obtained better scores than DK/DK2 on seven to ten FLPs whereas DK/DK2 outperformed Tam/Tam2 on four to six FLPs, the clustering method of Tam/Tam2 might be better than that of DK/DK2.

As for the difference between Tam2 and Kad, Kad may show superiority in the

	genCea/ P1.Pp50	genCea/ P1.Pp200	genCea/ P1.Pp1000	genTam/ P1.Pp50	genTam/ P1.Pp200	genTam/ P1.Pp1000	genKad/ P1.Pp50	genKad/ P1.Pp200	genKad/ P1.Pp1000
TL91-5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
TL91-6	0.00%	0.00%	0.00%	4.53%	4.53%	4.53%	2.72%	2.27%	0.00%
TL91-7	3.02%	1.34%	0.00%	30.49%	30.49%	30.49%	15.06%	15.86%	10.23%
TL91-8	3.17%	0.91%	0.00%	12.69%	10.11%	9.07%	7.24%	6.02%	5.75%
VCea91-10	23.57%	15.02%	13.47%	3.04%	2.98%	2.49%	1.41%	0.05%	0.00%
Kea91-11	3.76%	2.20%	0.00%	13.48%	13.41%	12.93%	3.99%	3.25%	1.93%
Kea91-11a	0.33%	0.00%	1.03%	10.11%	10.16%	9.66%	2.51%	2.85%	2.88%
TL91-12	6.80%	0.79%	0.00%	12.37%	9.33%	7.75%	6.36%	4.40%	4.85%
TL91-15	16.34%	13.44%	9.10%	5.67%	7.04%	2.69%	0.00%	3.01%	5.50%
Kea91-16	62.13%	57.23%	75.01%	7.99%	5.84%	5.54%	0.00%	4.92%	4.00%
Kea91-20	16.37%	17.08%	13.03%	0.41%	1.42%	1.79%	0.00%	0.00%	0.30%
TL91-20	13.79%	11.65%	24.32%	3.53%	5.28%	4.92%	1.03%	0.00%	2.27%
Tam92-20a	6.73%	14.67%	18.91%	2.05%	1.67%	2.13%	0.65%	1.27%	0.00%
TL91-30	18.91%	22.05%	107.44%	13.89%	10.78%	12.86%	7.32%	0.00%	2.52%
Tam92-30a	11.07%	7.96%	15.93%	8.88%	8.53%	8.21%	2.14%	0.00%	0.92%

Figure 6.4. Percentage by which each result is worse than the best for a particular problem (corresponding to Figure 6.3)

	genCea/ P1.Pp1000	genCea/ P4.Pp1000	genCea/ P10.Pp1000	genTam/ P1.Pp1000	genTam/ P4.Pp1000	genTam/ P10.Pp1000	genKad/ P1.Pp1000	genKad/ P4.Pp1000	genKad/ P10.Pp1000
TL91-5	228.15	228.15	228.15	228.15	228.15	228.15	228.15	228.15	228.15
TL91-6	361.45	361.45	361.45	377.84	377.84	377.84	361.45	366.37	368.01
TL91-7	595.89	606.18	611.58	777.57	777.57	777.57	656.83	662.47	669.98
TL91-8	883.76	890.57	910.04	963.95	968.52	963.95	934.56	949.24	955.49
VCea91-10	24440.66	26145.17	27898.12	22074.93	22047.65	22025.93	21538.61	21570.59	21470.67
Kea91-11	2947.11	2940.38	3056.54	3328.10	3328.10	3327.91	3004.08	3018.82	3074.64
Kea91-11a	2266.87	2277.56	2317.58	2460.45	2471.58	2463.23	2308.23	2313.26	2326.13
TL91-12	3614.52	3750.75	4094.22	3894.58	3873.53	3896.36	3789.74	3788.47	3785.82
TL91-15	8924.41	9987.69	10230.45	8400.31	8462.15	8320.92	8629.88	8503.98	8343.98
Kea91-16	114.09	120.80	121.69	68.80	65.40	67.40	67.80	67.40	68.40
Kea91-20	189.89	203.80	221.10	171.00	171.60	170.19	168.50	170.50	171.60
TL91-20	21266.95	24357.82	27848.71	17949.37	17932.51	17847.57	17495.76	17835.75	18293.26
Tam92-20a	25069.13	25126.82	25420.38	21532.31	21446.84	21656.46	21083.19	21582.12	21557.20
TL91-30	101773.54	116348.04	130610.35	55368.81	55923.73	56721.12	50300.66	51981.89	55603.00
Tam92-30a	52672.19	54657.95	56555.55	49162.65	50032.55	49927.20	45853.37	45999.94	47034.10

Figure 6.5. The mean of the best scores in the investigation for the number of populations (after 200000 evaluations)

	genCea/ P1.Pp1000	genCea/ P4.Pp1000	genCea/ P10.Pp1000	genTam/ P1.Pp1000	genTam/ P4.Pp1000	genTam/ P10.Pp1000	genKad/ P1.Pp1000	genKad/ P4.Pp1000	genKad/ P10.Pp1000
TL91-5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
TL91-6	0.00%	0.00%	0.00%	4.53%	4.53%	4.53%	0.00%	1.36%	1.81%
TL91-7	0.00%	1.73%	2.63%	30.49%	30.49%	30.49%	10.23%	11.17%	12.43%
TL91-8	0.00%	0.77%	2.97%	9.07%	9.59%	9.07%	5.75%	7.41%	8.12%
VCea91-10	13.83%	21.77%	29.94%	2.81%	2.69%	2.59%	0.32%	0.47%	0.00%
Kea91-11	0.23%	0.00%	3.95%	13.19%	13.19%	13.18%	2.17%	2.67%	4.57%
Kea91-11a	0.00%	0.47%	2.24%	8.54%	9.03%	8.66%	1.82%	2.05%	2.61%
TL91-12	0.00%	3.77%	13.27%	7.75%	7.17%	7.80%	4.85%	4.81%	4.74%
TL91-15	7.25%	20.03%	22.95%	0.95%	1.70%	0.00%	3.71%	2.20%	0.28%
Kea91-16	74.45%	84.71%	86.07%	5.20%	0.00%	3.06%	3.67%	3.06%	4.59%
Kea91-20	12.69%	20.95%	31.22%	1.48%	1.84%	1.00%	0.00%	1.19%	1.84%
TL91-20	21.55%	39.22%	59.17%	2.59%	2.50%	2.01%	0.00%	1.94%	4.56%
Tam92-20a	18.91%	19.18%	20.57%	2.13%	1.72%	2.72%	0.00%	2.37%	2.25%
TL91-30	102.33%	131.31%	159.66%	10.08%	11.18%	12.76%	0.00%	3.34%	10.54%
Tam92-30a	14.87%	19.20%	23.34%	7.22%	9.11%	8.88%	0.00%	0.32%	2.58%

Figure 6.6. Percentage by which each result is worse than the best for a particular problem (corresponding to Figure 6.5)

	oneCea/ P1.Pp1000	twoCea/ P1.Pp1000	genCea/ P1.Pp1000	oneTam/ P1.Pp1000	twoTam/ P1.Pp1000	genTam/ P1.Pp1000	oneKad/ P1.Pp1000	twoKad/ P1.Pp1000	genKad/ P1.Pp1000
TL91-5	228.15	228.15	228.15	228.15	228.15	228.15	228.15	228.15	228.15
TL91-6	365.38	365.38	361.45	377.84	377.84	377.84	376.20	372.92	361.45
TL91-7	623.60	621.79	595.89	777.57	777.57	777.57	679.02	670.17	656.83
TL91-8	900.35	892.93	883.76	963.95	963.95	963.95	958.78	963.95	934.56
VCea91-10	22144.52	22902.32	24440.66	22025.93	22025.93	22074.93	21883.44	21933.63	21538.61
Kea91-11	2941.81	2945.52	2947.11	3327.91	3327.91	3328.10	3151.11	3127.93	3004.08
Kea91-11a	2340.45	2329.21	2266.87	2460.45	2460.45	2460.45	2329.41	2318.58	2308.23
TL91-12	4198.62	4267.77	3614.52	3873.53	3873.53	3894.58	3763.17	3783.61	3789.74
TL91-15	10966.37	10609.23	8924.41	8282.39	8313.61	8400.31	8245.56	8267.09	8629.88
Kea91-16	134.80	135.69	114.09	64.80	64.00	68.80	64.00	65.59	67.80
Kea91-20	203.80	209.00	189.89	168.00	167.60	171.00	166.39	166.50	168.50
TL91-20	26533.39	28152.28	21266.95	17579.39	17464.25	17949.37	17104.42	17048.46	17495.76
Tam92-20a	25079.64	25124.67	25069.13	21387.11	21303.11	21532.31	21056.86	20894.95	21083.19
TL91-30	121340.15	126090.65	101773.54	53366.05	53396.93	55368.81	45828.25	45472.43	50300.66
Tam92-30a	54253.71	53851.98	52672.19	48907.71	48432.75	49162.65	44380.50	44436.23	45853.37

Figure 6.7. The mean of the best scores in reproduction investigation (after 200000 evaluations)

	oneCea/ P1.Pp1000	twoCea/ P1.Pp1000	genCea/ P1.Pp1000	oneTam/ P1.Pp1000	twoTam/ P1.Pp1000	genTam/ P1.Pp1000	oneKad/ P1.Pp1000	twoKad/ P1.Pp1000	genKad/ P1.Pp1000
TL91-5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
TL91-6	1.09%	1.09%	0.00%	4.53%	4.53%	4.53%	4.08%	3.17%	0.00%
TL91-7	4.65%	4.35%	0.00%	30.49%	30.49%	30.49%	13.95%	12.47%	10.23%
TL91-8	1.88%	1.04%	0.00%	9.07%	9.07%	9.07%	8.49%	9.07%	5.75%
VCea91-10	2.81%	6.33%	13.47%	2.26%	2.26%	2.49%	1.60%	1.83%	0.00%
Kea91-11	0.00%	0.13%	0.18%	13.12%	13.12%	13.13%	7.11%	6.33%	2.12%
Kea91-11a	3.25%	2.75%	0.00%	8.54%	8.54%	8.54%	2.76%	2.28%	1.82%
TL91-12	16.16%	18.07%	0.00%	7.17%	7.17%	7.75%	4.11%	4.68%	4.85%
TL91-15	33.00%	28.67%	8.23%	0.45%	0.83%	1.88%	0.00%	0.26%	4.66%
Kea91-16	110.63%	112.02%	78.27%	1.25%	0.00%	7.50%	0.00%	2.48%	5.94%
Kea91-20	22.48%	25.61%	14.12%	0.97%	0.73%	2.77%	0.00%	0.07%	1.27%
TL91-20	55.64%	65.13%	24.74%	3.11%	2.44%	5.28%	0.33%	0.00%	2.62%
Tam92-20a	20.03%	20.24%	19.98%	2.36%	1.95%	3.05%	0.77%	0.00%	0.90%
TL91-30	166.84%	177.29%	123.81%	17.36%	17.43%	21.76%	0.78%	0.00%	10.62%
Tam92-30a	22.25%	21.34%	18.68%	10.20%	9.13%	10.78%	0.00%	0.13%	3.32%

Figure 6.8. Percentage by which each result is worse than the best for a particular problem (corresponding to Figure 6.7)

	genCea/P1.Pp1000	genTam/P1.Pp1000
TL91-5	-	-
TL91-6	better	
TL91-7	better	
TL91-8	better	
VCea91-10		better
Kea91-11	better	
Kea91-11a	better	
TL91-12	better	
TL91-15		better
Kea91-16		better
Kea91-20		better
TL91-20		better
Tam92-20a		better
TL91-30		better
Tam92-30a		better

	twoCea/P1.Pp1000	twoTam/P1.Pp1000
TL91-5	-	-
TL91-6	better	
TL91-7	better	
TL91-8	better	
VCea91-10		better
Kea91-11	better	
Kea91-11a	better	
TL91-12		better
TL91-15		better
Kea91-16		better
Kea91-20		better
TL91-20		better
Tam92-20a		better
TL91-30		better
Tam92-30a		better

	genCea/P1.Pp1000	genDK/P1.Pp1000
TL91-5	better	
TL91-6	better	
TL91-7	better	
TL91-8	better	
VCea91-10		better
Kea91-11	better	
Kea91-11a	better	
TL91-12	better	
TL91-15	better	
Kea91-16		better
Kea91-20		better
TL91-20		better
Tam92-20a		better
TL91-30		better
Tam92-30a		better

	twoCea/P1.Pp1000	twoDK/P1.Pp1000
TL91-5	better	
TL91-6	better	
TL91-7	better	
TL91-8	better	
VCea91-10	better	
Kea91-11	better	
Kea91-11a	better	
TL91-12		better
TL91-15		better
Kea91-16		better
Kea91-20		better
TL91-20		better
Tam92-20a		better
TL91-30		better
Tam92-30a		better

Figure 6.9. The results of t-test between the Cea and Tam/DK algorithms

```
             genTam/P1.Pp1000   genDK/P1.Pp1000
-------------------------------------------------
   TL91-5       better
   TL91-6       -               -
   TL91-7                       better
   TL91-8       better
   VCea91-10    better
   Kea91-11                     better
   Kea91-11a    better
   TL91-12      better
   TL91-15      better
   Kea91-16     better
   Kea91-20     -               -
   TL91-20      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             twoTam/P1.Pp1000   twoDK/P1.Pp1000
-------------------------------------------------
   TL91-5       better
   TL91-6       -               -
   TL91-7                       better
   TL91-8       better
   VCea91-10    better
   Kea91-11                     better
   Kea91-11a    better
   TL91-12      better
   TL91-15      better
   Kea91-16     better
   Kea91-20                     better
   TL91-20      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             genTam2/P1.Pp1000  genDK2/P1.Pp1000
-------------------------------------------------
   TL91-5       better
   TL91-6       better
   TL91-7                       better
   TL91-8       better
   VCea91-10                    better
   Kea91-11     better
   Kea91-11a    better
   TL91-12      better
   TL91-15      better
   Kea91-16     better
   Kea91-20     better
   TL91-20      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             twoTam2/P1.Pp1000  twoDK2/P1.Pp1000
-------------------------------------------------
   TL91-5       better
   TL91-6       -               -
   TL91-7                       better
   TL91-8       -               -
   VCea91-10    better
   Kea91-11     better
   Kea91-11a    better
   TL91-12                      better
   TL91-15      better
   Kea91-16     -               -
   Kea91-20     -               -
   TL91-20      better
   Tam92-20a    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

Figure 6.10. The results of t-test between the Tam/Tam2 and DK/DK2 algorithms

```
            genTam2/P1.Pp1000  genKad/P1.Pp1000
-----------------------------------------------
 TL91-5      -                  -
 TL91-6      better
 TL91-7      better
 TL91-8      -                  -
 VCea91-10                      better
 Kea91-11    better
 Kea91-11a   -                  -
 TL91-12     better
 TL91-15     better
 Kea91-16    better
 Kea91-20    -                  -
 TL91-20                        better
 Tam92-20a                      better
 TL91-30                        better
 Tam92-30a                      better
-----------------------------------------------
```

```
                        twoTam2/P1.Pp1000  twoKad/P1.Pp1000
-----------------------------------------------
 TL91-5      -                  -
 TL91-6                         better
 TL91-7                         better
 TL91-8      better
 VCea91-10                      better
 Kea91-11    better
 Kea91-11a                      better
 TL91-12                        better
 TL91-15     -                  -
 Kea91-16    -                  -
 Kea91-20    -                  -
 TL91-20                        better
 Tam92-20a                      better
 TL91-30                        better
 Tam92-30a                      better
-----------------------------------------------
```

Figure 6.11. The results of t-test between the Tam2 and Kad algorithms

two reproduction method as shown in Figure 6.11, but the cause of the difference has not been clear so far.

As Figure 6.12 suggests, the new algorithms, Tam2/DK2, certainly improved the performance of the replicated algorithms, Tam/DK, in many FLPs, respectively. However, as shown in Figure 6.13, the Cea algorithm still showed better performance on smaller FLPs whereas Tam2/DK2/Kad performed better on larger FLPs. That is, the relations between Cea and the new algorithms, Tam2/DK2/Kad, are still similar to those between Cea and the replicated algorithms, Tam/DK, shown in Figure 6.9. Again, although one or two of fifteen t-tests may lead wrong results, the tendency that Cea is suitable for smaller FLPs and that Tam2/DK2/Kad is strong for larger FLPs may be too clear to be rejected. Consequently, the number of facilities may significantly influence GA performance on FLPs.

Comparison of GAs with Other Algorithms As shown in Figure 6.14, the original papers' experimental conditions are quite different. So, in order to do fair comparison, I used the basis as shown in Table 6.17.

The compared results are summarised in Figure 6.15. Here, the results in Kea91-11, 11a, 16 and 20 are used for the comparison with [KJK91]'s algorithm (simulated annealing: SA); those in TL91-5, 6, 7, 8, 12, 15, 20 and 30 are used for the comparison with [TL91]'s algorithm (quasi-Newton method: QN); those

```
             genTam/P1.Pp1000   genTam2/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6                       better
   TL91-7                       better
   TL91-8                       better
   VCea91-10                    better
   Kea91-11                     better
   Kea91-11a                    better
   TL91-12                      better
   TL91-15                      better
   Kea91-16                     better
   Kea91-20                     better
   TL91-20                      better
   Tam92-20a                    better
   TL91-30     better
   Tam92-30a                    better
-------------------------------------------------
```

```
             twoTam/P1.Pp1000   twoTam2/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6                       better
   TL91-7                       better
   TL91-8       better
   VCea91-10                    better
   Kea91-11                     better
   Kea91-11a                    better
   TL91-12                      better
   TL91-15                      better
   Kea91-16     -               -
   Kea91-20     -               -
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             genDK/P1.Pp1000    genDK2/P1.Pp1000
-------------------------------------------------
   TL91-5                       better
   TL91-6                       better
   TL91-7                       better
   TL91-8                       better
   VCea91-10                    better
   Kea91-11                     better
   Kea91-11a                    better
   TL91-12                      better
   TL91-15                      better
   Kea91-16                     better
   Kea91-20     -               -
   TL91-20                      better
   Tam92-20a                    better
   TL91-30     better
   Tam92-30a                    better
-------------------------------------------------
```

```
             twoDK/P1.Pp1000    twoDK2/P1.Pp1000
-------------------------------------------------
   TL91-5                       better
   TL91-6                       better
   TL91-7                       better
   TL91-8                       better
   VCea91-10                    better
   Kea91-11                     better
   Kea91-11a                    better
   TL91-12                      better
   TL91-15                      better
   Kea91-16     better
   Kea91-20     -               -
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

Figure 6.12. The results of t-test between the Tam/DK and Tam2/DK2 algorithms

```
             genCea/P1.Pp1000   genTam2/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6       better
   TL91-7       better
   TL91-8       better
   VCea91-10                    better
   Kea91-11     -               -
   Kea91-11a    better
   TL91-12      better
   TL91-15                      better
   Kea91-16                     better
   Kea91-20                     better
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             twoCea/P1.Pp1000   twoTam2/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6       better
   TL91-7       better
   TL91-8       better
   VCea91-10                    better
   Kea91-11     better
   Kea91-11a    better
   TL91-12                      better
   TL91-15                      better
   Kea91-16                     better
   Kea91-20                     better
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             genCea/P1.Pp1000   genDK2/P1.Pp1000
-------------------------------------------------
   TL91-5       better
   TL91-6       better
   TL91-7       better
   TL91-8       better
   VCea91-10                    better
   Kea91-11     better
   Kea91-11a    better
   TL91-12      better
   TL91-15                      better
   Kea91-16                     better
   Kea91-20                     better
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             twoCea/P1.Pp1000   twoDK2/P1.Pp1000
-------------------------------------------------
   TL91-5       better
   TL91-6       better
   TL91-7       better
   TL91-8       better
   VCea91-10                    better
   Kea91-11     better
   Kea91-11a    better
   TL91-12                      better
   TL91-15                      better
   Kea91-16                     better
   Kea91-20                     better
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             genCea/P1.Pp1000   genKad/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6       better
   TL91-7       better
   TL91-8       better
   VCea91-10                    better
   Kea91-11     better
   Kea91-11a    better
   TL91-12      better
   TL91-15                      better
   Kea91-16                     better
   Kea91-20                     better
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             twoCea/P1.Pp1000   twoKad/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6       better
   TL91-7       better
   TL91-8       better
   VCea91-10                    better
   Kea91-11     better
   Kea91-11a                    better
   TL91-12                      better
   TL91-15                      better
   Kea91-16                     better
   Kea91-20                     better
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

Figure 6.13. The results of t-test between the Cea and Tam2/DK2/Kad algorithms

problem	best score	mean score	standard deviation	the number of samples	the number of evaluations (etc.)
Kea91-11	2829.4	2829.4	0	1	-
Kea91-11a	2287.041	2287.041	0	1	-
Kea91-16	64	83.5	11.517	10	192200
Kea91-20	153	170.75	13.679	10	219024
TL91-5	246.82	246.82	0	1	CPU time = 0.32sec
TL91-6	514	514	0	1	CPU time = 0.57sec
TL91-7	559	559	0	1	CPU time = 4.50sec
TL91-8	839	839	0	1	CPU time = 12.45sec
TL91-12	3162	3162	0	1	CPU time = 89.50sec
TL91-15	5862	5862	0	1	CPU time = 379.63sec
Tam92-20a	23544	23544	0	1	1596
Tam92-30a	45044	45044	0	1	3712
VCea91-10	24445	24445	0	1	CPU time = 847sec

# The result on Kea91-16 is retrieved from the case of the number of populations = 8.
# The CPU times on TL91-5,6,7,8,12 and 15 are on CRAY MP/SE supercomputer.
# The CPU time on VCea91-10 is on Apollo DN3500 (5MIPS).
# TL91-20 and TL91-30 were not able to be solved by [TL91]'s algorithm.

Figure 6.14. The scores reported in previous papers

in Tam92-20a and Tam92-30a are used for the comparison with [Tam92b]'s algorithm (SA); and those in VCea91-10 are used for the comparison with [VCCV91]'s algorithm (QN).

Since the confidence of each t-test is 95%, we have to be careful that 5% of t-test results in the figure are probably wrong. Therefore, I will claim the superiority of a particular algorithm only when almost all t-tests show coherent tendency.

Regarding [KJK91]'s algorithm, many better results appeared by the Cea and the Tam2/DK2/Kad algorithms in Kea91-11 and Kea91-11a. In Kea91-16, the Cea algorithm showed worse performance whereas all the other algorithms got better results. In Kea91-20, all Genitor reproduction methods with Tam/DK and Tam2/DK2/Kad showed better results while any reproduction methods with Cea were worse. Consequently, it may be difficult to say whether [KJK91]'s algorithm is better or worse than Cea and Tam/DK algorithms. In contrast, Tam2/DK2/Kad may be better than [KJK91]'s especially in Genitor reproduction.

As regards [TL91]'s algorithm, the performances were clearly classified into three types from the number of facilities. That is, almost all GAs showed better results in TL91-5 and 6; all GAs did not show better results in TL91-7, 8, 12

Table 6.17. How to compare GAs with other algorithms

- If the original paper included ten samples, the t-test between the original paper's results and the experiments on a GA will be done. At that, the criterion of significance of 0.05 will be used. After the t-test, if the GA indicates better/worse performance, the word, "better/worse", will be indicated in the corresponding place in Figure 6.15.
  Otherwise, "-" will be indicated.

- If the original paper included only one sample, the sample will be compared with the best score of ten experiments of a GA. Then, if the GA's performance is better, the word, "better", will be put in Figure 6.15.
  Otherwise, "-" will be indicated.

- As for the number of evaluations, the result of 200000 evaluations are used for the comparison except for Tam92-20a and Tam92-30a.
  For Tam92-20a and Tam92-30a, the result of 2000 and 4000 evaluations will be used, respectively so that the number of evaluations can be as with original paper's.

	genCea/ P1.Pp50	genCea/ P1.Pp200	genCea/ P1.Pp1000	genCea/ P4.Pp1000	genCea/ P10.Pp1000	genCea/ P1.Ppbest	oneCea/ P1.Pp1000	twoCea/ P1.Pp1000
Kea91-11	-	-	-	better	-	-	better	-
Kea91-11a	better	better	better	better	better	better	better	better
Kea91-16	worse	worse	worse	worse	worse	worse	worse	worse
Kea91-20	worse	worse	worse	worse	worse	worse	worse	worse
TL91-5	better	better	better	better	better	better	better	better
TL91-6	better	better	better	better	better	better	better	better
TL91-7	-	-	-	-	-	-	-	-
TL91-8	-	-	-	-	-	-	-	-
TL91-12	-	-	-	-	-	-	-	-
TL91-15	-	-	-	-	-	-	-	-
TL91-20								
TL91-30								
Tam92-20a*	-	-	-	-	-	-	-	-
Tam92-30a**	-	-	-	-	-	-	-	-
VCea91-10	better	better	better	better	-	better	better	better

	genTam/ P1.Pp50	genTam/ P1.Pp200	genTam/ P1.Pp1000	genTam/ P4.Pp1000	genTam/ P10.Pp1000	genTam/ P1.Ppbest	oneTam/ P1.Pp1000	twoTam/ P1.Pp1000	genDK/ P1.Pp1000	genDK/ P1.Ppbest	twoDK/ P1.Pp1000
Kea91-11	-	-	-	-	-	-	-	-	-	-	-
Kea91-11a	-	-	-	-	-	-	-	-	-	-	-
Kea91-16	better	better	better	better	better	better	better	better	better	better	better
Kea91-20	-	-	-	-	-	-	better	better	-	worse	better
TL91-5	better	better	better	better	better	better	better	better	-	-	-
TL91-6	better	better	better	better	better	better	better	better	better	better	better
TL91-7	-	-	-	-	-	-	-	-	-	-	-
TL91-8	-	-	-	-	-	-	-	-	-	-	-
TL91-12	-	-	-	-	-	-	-	-	-	-	-
TL91-15	-	-	-	-	-	-	-	-	-	-	-
TL91-20											
TL91-30											
Tam92-20a*	better	better	better	-	better	better	better	better	better	better	better
Tam92-30a**	-	-	-	-	-	-	-	-	-	-	-
VCea91-10	better	better	better	better	better	better	better	better	better	better	better

	genKad/ P1.Pp50	genKad/ P1.Pp200	genKad/ P1.Pp1000	genKad/ P4.Pp1000	genKad/ P10.Pp1000	oneKad/ P1.Pp1000	twoKad/ P1.Pp1000	genTam2/ P1.Pp1000	twoTam2/ P1.Pp1000	genDK2/ P1.Pp1000	twoDK2/ P1.Pp1000
Kea91-11	-	-	better	better	-	-	-	better	better	-	-
Kea91-11a	better	better	-	-	-	-	-	-	-	better	better
Kea91-16	better	better	better	better	better	better	better	better	better	better	better
Kea91-20	-	-	-	-	-	better	better	-	better	-	better
TL91-5	better	better	better	better	better	better	better	better	better	better	better
TL91-6	better	better	better	better	better	better	better	better	better	better	better
TL91-7	-	-	-	-	-	-	-	-	-	-	-
TL91-8	-	-	-	-	-	-	-	-	-	-	-
TL91-12	-	-	-	-	-	-	-	-	-	-	-
TL91-15	-	-	-	-	-	-	-	-	-	-	-
TL91-20											
TL91-30											
Tam92-20a*	better	-	better	-	-	better	better	better	better	better	better
Tam92-30a**	-	-	-	-	-	-	-	-	-	-	-
VCea91-10	better	better	better	better	better	better	better	better	better	better	better

The result of 200000 evaluations are used for the comparison except for the following cases.
* For Tam92-20a, the result of 2000 evaluations are used.
** For Tam92-30a, the result of 4000 evaluations are used.

Figure 6.15. The result of comparing GAs with other algorithms

and 15; and all GAs were able to obtain some solutions in TL91-20 and 30. Accordingly, the [TL91] algorithm's effective range may be much narrower than that of GAs.

Comparing GAs with [Tam92b]'s results, many better layouts were generated in Tam92-20a with Tam/DK and Tam2/DK2/Kad algorithms. On the other hand, no better results were found in Tam92-30a after 4000 evaluations. However, as shown in Appendix B, all the GAs except for Cea obtained better layouts in Tam92-20a and all Genitor representation with Tam2/DK2/Kad reached better layouts in Tam92-30a, after 200000 evaluations. Hence, [Tam92b] can be said to be a quicker algorithm than GAs, though it might cause premature convergence.

In VCea91-10, only one GA failed to show better performance than the results there. Therefore, GAs are generally superior to [VCCV91]'s algorithm.

In conclusion, compared with simulated annealing (i.e. the algorithms in [KJK91] and [Tam92b]), the Cea algorithm may not be so good; Tam/DK may be as good as SA; and Tam2/DK2/Kad may be better especially with Genitor reproduction methods. As for quasi-Newton methods (i.e. the algorithms in [TL91] and [VCCV91]), GAs are generally better than QN methods except for the range where [TL91]'s algorithm shows remarkable performance.

Population Size Effects To see the population size effects on Cea, Tam and Kad algorithms, F-tests are first used. As shown in Figure 6.16, the population size seems to be influential factor for the Cea algorithm because Cea's performance often varied under different population size. On the other hand, Tam and Kad algorithms were less affected by different population size.

Then, to observe the relation between the Cea algorithm and population size, protected t-tests are used as shown in Figure 6.17. Although the tendency is not so strong, large population size may be preferable for the Cea algorithm.

However, as many graphs in Appendix D suggests, the convergence speed of GAs with smaller population size may be faster.

Effects of Number of Populations As shown in Figures 6.18, the performance of Cea is often affected by the number of populations whereas those of Tam and Kad are less influenced. Using protected t-tests, Cea's tendency that

```
           | genCea  | genTam  | genKad  |
------------------------------------------
     TL91-5|  0.00   |  0.00   |  0.00   |
     TL91-6|  0.00   |  0.00   |  5.69*  |
     TL91-7| 11.68*  |  0.00   |  3.77*  |
     TL91-8| 19.38*  |  9.49*  |  3.81*  |
  VCea91-10|  4.26*  |  1.96   |  1.13   |
   Kea91-11| 11.12*  |  2.83   |  1.61   |
  Kea91-11a|  1.91   | 13.72*  |  0.48   |
    TL91-12| 14.30*  |  5.84*  |  0.93   |
    TL91-15|  7.44*  |  2.77   |  5.28*  |
   Kea91-16| 17.08*  |  0.25   |  1.26   |
   Kea91-20|  1.21   |  1.79   |  0.04   |
    TL91-20| 12.43*  |  0.65   |  2.12   |
  Tam92-20a| 54.10*  |  0.52   |  1.51   |
    TL91-30| 93.09*  |  2.00   | 17.61*  |
  Tam92-30a|  7.48*  |  0.85   | 10.35*  |
------------------------------------------

"*" indicates significant difference was found.
```

Figure 6.16. The results of F-tests for population size investigation

	genCea/P1.Pp1000	genCea/P1.Pp200
TL91-5	-	-
TL91-6	-	-
TL91-7	-	-
TL91-8	better	
VCea91-10	better	
Kea91-11	-	-
Kea91-11a	-	-
TL91-12	better	
TL91-15	-	-
Kea91-16		better
Kea91-20	-	-
TL91-20		better
Tam92-20a		better
TL91-30		better
Tam92-30a		better

	genCea/P1.Pp200	genCea/P1.Pp50
TL91-5	-	-
TL91-6	-	-
TL91-7	-	-
TL91-8	-	-
VCea91-10	-	-
Kea91-11	better	
Kea91-11a	-	-
TL91-12	-	-
TL91-15	better	
Kea91-16	better	
Kea91-20	-	-
TL91-20	better	
Tam92-20a		better
TL91-30	better	
Tam92-30a	better	

Figure 6.17. The results of protected t-tests for population size investigation

	genCea	genTam	genKad
TL91-5	0.00	0.00	0.00
TL91-6	0.00	0.00	2.60
TL91-7	9.55*	0.00	0.32
TL91-8	10.60*	1.00	7.31*
VCea91-10	10.32*	7.56*	0.10
Kea91-11	7.75*	0.50	2.00
Kea91-11a	6.23*	14.63*	8.24*
TL91-12	21.59*	0.50	0.00
TL91-15	26.44*	1.38	2.46
Kea91-16	7.36*	1.85	0.11
Kea91-20	20.45*	1.33	2.48
TL91-20	28.85*	0.10	7.03*
Tam92-20a	3.29	1.51	9.25*
TL91-30	21.00*	1.26	20.15*
Tam92-30a	31.33*	8.11*	15.59*

"*" indicates significant difference was found.

Figure 6.18. The results of F-tests for the investigation of the number of populations

smaller number of populations may be more suitable is observed as shown in Figures 6.19.

Reproduction Methods As shown in Figure 6.20, no typical difference can be seen in any three algorithms regarding the results after 200000 evaluations, However, the convergence speed may be substantially different as many graphs in Appendix F suggests. That is, two reproduction (Genitor with twin children) is the fastest and gen reproduction (generation based) is the slowest. Considering the fact that these three methods' final results are similar in quality, Genitor reproduction especially with twin children may be a more effective GA than the generation based approach.

Effect of Tuned Parameters In this experiment, tuned GA parameters which showed the best performance for a particular problem under a particular GA are used as well as other arbitrarily chosen parameters. However, as shown in Figure 6.15 and in Appendix B, the performance of tuned parameters only showed similar tendency. Therefore, I would like to claim that GAs may be quite robust for the choice of GA parameters.

	genCea/P1.Pp1000	genCea/P4.Pp1000
TL91-5	-	-
TL91-6	-	-
TL91-7	-	-
TL91-8	better	
VCea91-10	-	-
Kea91-11	better	
Kea91-11a	better	
TL91-12	better	
TL91-15	-	-
Kea91-16	-	-
Kea91-20	-	-
TL91-20	-	-
Tam92-20a	-	-
TL91-30	-	-
Tam92-30a	-	-

	genCea/P4.Pp1000	genCea/P10.Pp1000
TL91-5	-	-
TL91-6	-	-
TL91-7	better	
TL91-8	-	-
VCea91-10	-	-
Kea91-11	-	-
Kea91-11a	-	-
TL91-12	-	-
TL91-15	better	
Kea91-16	better	
Kea91-20	-	-
TL91-20	-	-
Tam92-20a	-	-
TL91-30	-	-
Tam92-30a	-	-

Figure 6.19. The results of protected t-tests for the investigation of the number of populations

6.4 Summary

In conclusion, I obtained the following results.

As regards the performance comparison between GAs and other algorithms, I obtained a result showing that the performance of GAs is generally better than those of previously reported Simulated Annealing (SA) and quasi-Newton (QN) methods on the standard FLPs. This superiority was especially significant in Tam2, DK2, and Kad algorithms when they use Genitor reproduction producing two complementary children.

From the comparison of GAs with each other, it was observed that the algorithm's suitability may be highly dependent on the number of facilities. This is because while the Cea algorithm outperformed others in almost all of those FLPs having at most twelve facilities, the other five GAs clearly showed better performance on FLPs including at least fifteen facilities. Since the five algorithms put a limitation on initial search space by some clustering method, this sort of space limitation may be effective for FLPs consisting of many facilities. However, both the Tam and DK algorithms often fell into premature convergence which led to poor solutions. Considering the fact that these algorithms have strict limitation for search space, such limitation may not be a good idea generally in FLPs.

```
             genCea/P1.Pp1000   twoCea/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6       better
   TL91-7       better
   TL91-8       better
   VCea91-10                    better
   Kea91-11     -               -
   Kea91-11a    better
   TL91-12      better
   TL91-15      better
   Kea91-16     better
   Kea91-20     better
   TL91-20      better
   Tam92-20a    better
   TL91-30      better
   Tam92-30a    better
-------------------------------------------------
```

```
             oneCea/P1.Pp1000   twoCea/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6       -               -
   TL91-7       -               -
   TL91-8                       better
   VCea91-10    better
   Kea91-11     -               -
   Kea91-11a                    better
   TL91-12      better
   TL91-15                      better
   Kea91-16     -               -
   Kea91-20     better
   TL91-20      better
   Tam92-20a    better
   TL91-30      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             genTam/P1.Pp1000   twoTam/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6       -               -
   TL91-7       -               -
   TL91-8       better
   VCea91-10                    better
   Kea91-11     -               -
   Kea91-11a    -               -
   TL91-12                      better
   TL91-15                      better
   Kea91-16                     better
   Kea91-20                     better
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             oneTam/P1.Pp1000   twoTam/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6       -               -
   TL91-7       -               -
   TL91-8       better
   VCea91-10    better
   Kea91-11     better
   Kea91-11a    -               -
   TL91-12      -               -
   TL91-15      better
   Kea91-16     -               -
   Kea91-20     -               -
   TL91-20                      better
   Tam92-20a                    better
   TL91-30      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             genKad/P1.Pp1000   twoKad/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6       better
   TL91-7       better
   TL91-8       better
   VCea91-10    better
   Kea91-11     better
   Kea91-11a    better
   TL91-12      -               -
   TL91-15                      better
   Kea91-16                     better
   Kea91-20                     better
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a                    better
-------------------------------------------------
```

```
             oneKad/P1.Pp1000   twoKad/P1.Pp1000
-------------------------------------------------
   TL91-5       -               -
   TL91-6                       better
   TL91-7                       better
   TL91-8       better
   VCea91-10    better
   Kea91-11                     better
   Kea91-11a                    better
   TL91-12      better
   TL91-15      better
   Kea91-16     better
   Kea91-20     -               -
   TL91-20                      better
   Tam92-20a                    better
   TL91-30                      better
   Tam92-30a    better
-------------------------------------------------
```

Figure 6.20. The results of t-test for reproduction investigation

Extensive details of the results appear is appendices B to G. The contents are as follows.

- Appendix B: Tables showing the performance of each GA method in each of the fifteen test problems.
- Appendix C: Score/time graphs for all six GA methods, for different problems and parameter variations.
- Appendix D: Score/time graphs for different population sizes, for each problem and with other GA parameters fixed.
- Appendix E: As Appendix D, but for number of populations rather than population size.
- Appendix F: As Appendix D, but for reproduction method rather than population size.
- Appendix G: Diagram of the best layout found by the GA for each problem.

Regarding the investigation of GA parameters, I could confirm that higher mutation rate may improve the performance of the Cea algorithm. Taking account of the characteristic that the Cea algorithm starts the search from mediocre solutions and it needs a high mutation rate, this result may be reasonable. Also, I confirmed that Genitor reproduction especially for producing two children was generally better than generation-based reproduction. In other words, the results observed in this experiment are consistent with other GA studies in fields different from FLPs as shown in [Dav91].

Chapter 7

Conclusions

7.1 Summary of My Research Work

In this thesis, facility layout problems (FLPs) and Genetic Algorithms (GAs) were reviewed. I argued that suboptimal approaches are suitable for FLPs owing to their NP-completeness, and that GAs, which have been recognised as a useful method for NP-complete problems, may be suitable for FLPs. I then noted that various researchers have recently investigated the application of GAs to FLPs. However, the FLPs in question were usually identical FLPs. I conjectured that there is also a need to examine the use of GAs on non-identical FLPs, which are more difficult than identical FLPs and more typical of real problems. I also noted that GA/FLP researchers so far had not conducted any significant investigation of the effects of varying GA parameters. Also, it is notable that several test problems are referred to in different papers, but few of these problems were examined in any two different investigations.

With all this in mind, I established the following immediate research interests:

- Investigating GA parameters on a varied set of FLPs
- Comparing GA performance with other methods on a varied set of FLPs
- Comparing possible GAs with each other on a varied set of FLPs

To research the above topics, I initially established fifteen standard FLPs. These problems were obtained from previous papers which worked on non-identical

FLPs with various algorithms. Therefore, results exist for these problems arising from various methods, constituting a useful basis for further comparative study.

Then, in the light of a literature review, I suggested Slicing Tree Structures (STSs) may be a reasonable representation of physical layouts. In addition to explaining the concept of STS, I described two types of methods by which an STS can correspond to a physical layout, and discussed aspects of the FLP search space from the perspective of STSs.

After this, I set up six types of GAs (Cea, Tam, DK, Tam2, DK2, Kad) which can solve FLPs using an STS-based representation. The Cea and Tam algorithms are duplicates of the GAs used in [CHMR91] and [Tam92a]. While the Cea algorithm searches all possible layouts which can be expressed by STSs, the Tam algorithm first limits the search space using the result of a clustering method and then searches for solutions within the limited search space. The DK algorithm is the same as the Tam algorithm except that DK uses another clustering method, described in [DK85], to limit the search space.

In contrast, the Tam2 and DK2 algorithms relax the search space limitations of the Tam and DK algorithms, respectively. These two algorithms start the search within the limited space like Tam and DK, but they are allowed to widen the search space by mutations which allow them to escape the initial limitations. The Kad algorithm is a hybrid of Tam2 and DK2, since it starts the search from both of their initial limited spaces. In common with Tam2 and DK2, the Kad algorithm can widen the search space by mutations to escape the initial limitations.

In experiments I investigated GA parameter effects and compared the GAs' performance not only with each other but also with other algorithms. As regards performance comparison, I obtained a result suggesting that the performance of GAs is generally better than those of previously reported Simulated Annealing (SA) and quasi-Newton (QN) methods on the standard FLPs. This superiority was especially significant with the Tam2, DK2, and Kad algorithms when they used Genitor reproduction producing two complementary children.

From the comparison of different GAs with each other, it was observed that the algorithms' suitability was highly dependent on the number of facilities. While the Cea algorithm outperformed others in almost all of those FLPs having at most twelve facilities, the other five GAs clearly showed better performance on FLPs including at least fifteen facilities. This suggests that initial limitation of the search using a clustering method is generally good for larger problems. Further, since Tam2 and DK2 generally outperformed Tam and DK, we can conclude that initial focusing on an area of the search by clustering is most effective when coupled with unrestricted search in the initial determined region.

The following comments summarise the main effects observed in the investigation of GA parameters. First, the Cea algorithm performs best with relatively high mutation rates. This is reasonable, since the initial mediocre STS topology in Cea can be altered only by a mutation operator (MU3). Second, solution quality generally improves as population size increases, although at the expense of slower convergence times. Third, Genitor-style (steady state) reproduction was better than generational reproduction. On the whole, experience here is consistent with GA studies in general.

7.2 Suggestions for Future Work

Though many things were observed through my work, further research might uncover more interesting points. Here, I will mention some points as possible future work.

Other Possible Investigations of GAs First, as mentioned in Section 5.4, stochastic clustering methods may be valuable for investigation. This is because those methods will generate various kinds of clustering results which may effectively scatter initial chromosomes over the search space. Starting from good solutions, GAs might find good solutions more quickly.

Second, although the Cea algorithm showed good performance on FLPs consisting of small numbers of facilities, I analysed it only from the perspective of the search space limitation. Because the Cea algorithm uses unique crossover and mutation methods, these influences could be examined more closely.

Third, more investigation of GA parameters, especially selection methods, could be done. Although I obtained a result showing that tournament selection methods worked well in the Tam and DK algorithms, I have not yet investigated further. It might be interesting to compare various selection methods including the tournament selection of different tournament sizes.

Other Algorithms Since I used only non-identical FLPs as benchmark problems, I only performed comparative study with algorithms which have been previously applied to such non-identical FLPs. This enabled me to refer to previously recorded results. However, there are other algorithms which may be good on non-identical FLPs, but so far have been tested only on identical FLPs. An example is [YP93]'s parallel simulated evolution method. Further work might involve replicating the method and applying it to the test problems used here.

Other Representations than STSs Although I suggested that STSs may be the most suitable representation so far for FLPs, STSs have some defects as mentioned in Section 4.1. For example, the current STSs can represent only layouts consisting of rectangles whose orientations follow the perpendicular or horizontal axis. Because there may be many cases where facilities should be oriented in other ways (e.g. diagonally), other possible representation techniques for FLPs should be developed.

Penalty Functions for Layouts In my implementation, only aspect ratio limitations and the minimum rectangular area that encloses all facilities were taken into account as penalty factors of layouts. Nevertheless, it is obvious that many other types of penalty functions should be considered. For instance, if a facility is assigned to a region having an unnatural shape, such as an L-shape, it should be penalised because it may make the resulting layout physically impractical. Actually, my algorithms sometimes produced such cases. As shown in Figure 7.1, the best result obtained in the Tam92-20a problem assigned facility No.3 to a region including a very narrow part which is hardly useful, and assigned facility No.6 to a place consisting of two separate parts with a prespecified area located between them. Accordingly, many extra types of penalty functions will be necessary for practical applications.

But, simple mixture of various penalty functions might not be an immediate

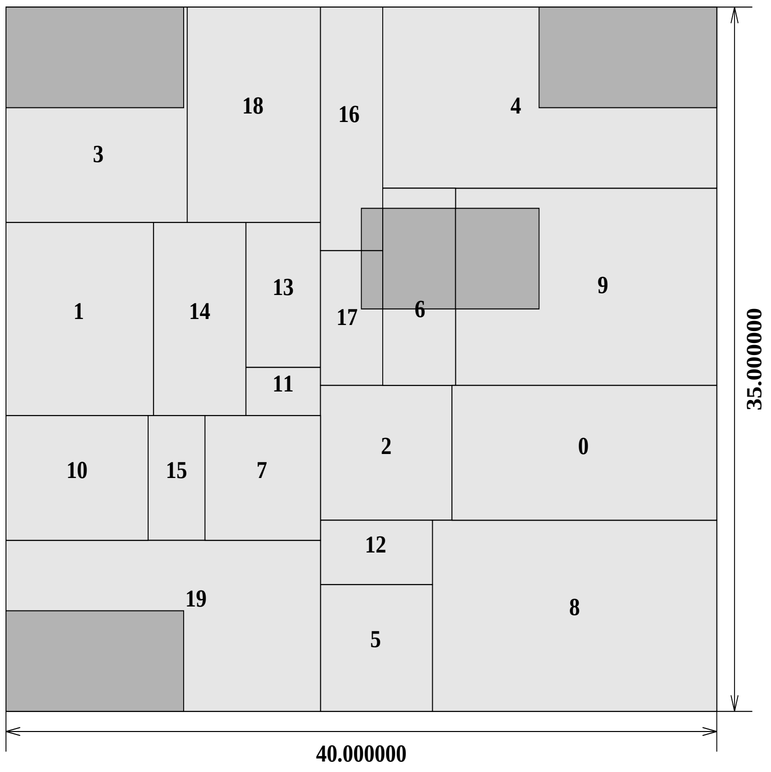


Figure 7.1. A difficult layout for practical use

solution because the factors of penalty functions may greatly influence the behaviour of GAs. For example, when I tackled Kea91-16 and Kea91-20 using high penalty values for the violation of aspect ratio limitation, I scarcely reached good solutions in Kea91-20, though I easily reached the ideal solution in Kea91-16. So, the factor of each penalty function should be carefully chosen. However, it may be highly dependent on each FLP specifications; therefore, some dynamic change of the penalty factors such as used in [ST93] might be a good approach.

An alternative to using penalty functions is to perform Pareto-dominance based multi-objective optimisation (e.g. [HN93]). In this case, penalty factors do not need to be chosen at all. A recent study, for example, has shown this to be a successful approach to pipe-choice optimisation.

Adaptive Interpretations for STS operators Instead of giving penalty to a physical layout including non-rectangular regions, adaptive interpretations for STS operators may be worth considering. As already mentioned in Section 4.1, STSs generally include four types of operators, `U`, `B`, `R` and `L`, and their interpretations are fixed. But, the interpretation of these STS operators could be adaptively changed so that all the facility regions can have rectangular or almost rectangular shapes.

For example, whereas the ordinary 'static' interpretation of `B` is "Put facility x below facility y", the adaptive interpretation of `B` may mean "Arrange facilities x and y so as to best avoid various problems with prespecified areas; all else being the same, use the static interpretation." So, for instance, the layout shown in Figure 7.2(a), which includes a typical non-rectangular region for facility No.3, could be interpreted as shown in Figure 7.2(b). This is because, if the operator `B` in the STS is interpreted as another operator, e.g. `R`; then the physical layout might be comprised of rectangular or almost rectangular regions only. So, in such a case, by using adaptive interpretations for STS operators, not only less suitable layouts can be excluded but also more suitable layouts can be discovered.

Related to this issue, another possible approach can be mentioned. For example, the meanings of operators `U` and `B` and/or `R` and `L` may be swapped over, respectively. That is, if the interpretations of `U` and `B` are exchanged, the whole layout will be almost upside down and no strange shaped region caused by the prespecified areas might not appear as shown in Figure 7.2(c).

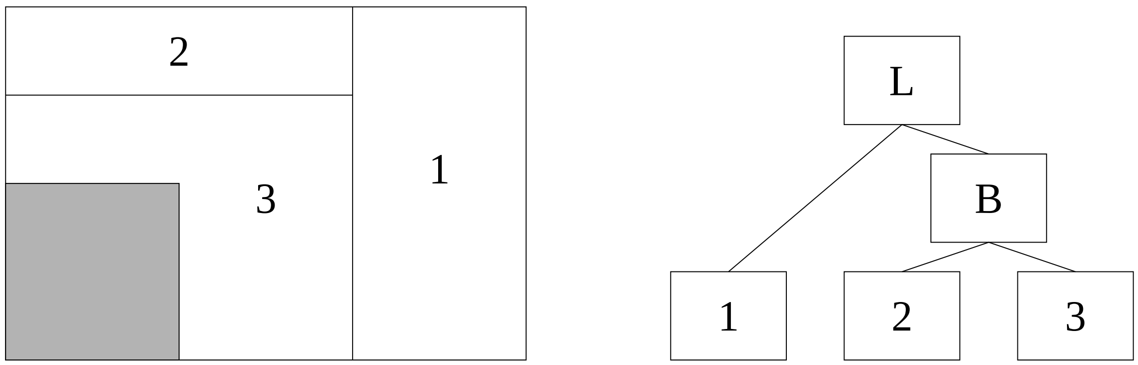
However, as there are many sorts of possible approaches like these, how to adapt each STS operator's interpretation may be an issue for the further research.

Indicator of the difficulty of FLPs In this study, we have only distinguished between different FLPs in the basis of the number of facilities. This is not, however, a necessarily good indicator of the difficulty of the problems. A different aspect of FLPs worth considering is the character of the traffic matrix. For example, an FLP with thousands of facilities will actually be very easy to solve if only a small handful of facilities have non-zero traffic between them. In general, study of FLP difficulty with regard to aspects of the traffic matrix will be useful.

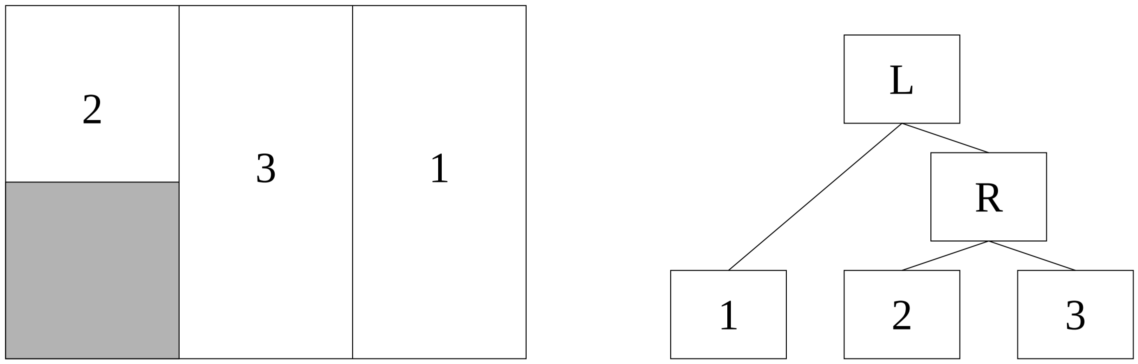
7.3 Final Comments

Though many interesting points for future work were left undone, I think my research interests set at the beginning of this work have been sufficiently investigated. Fortunately, the strength of GAs for FLPs was confirmed, as I expected.

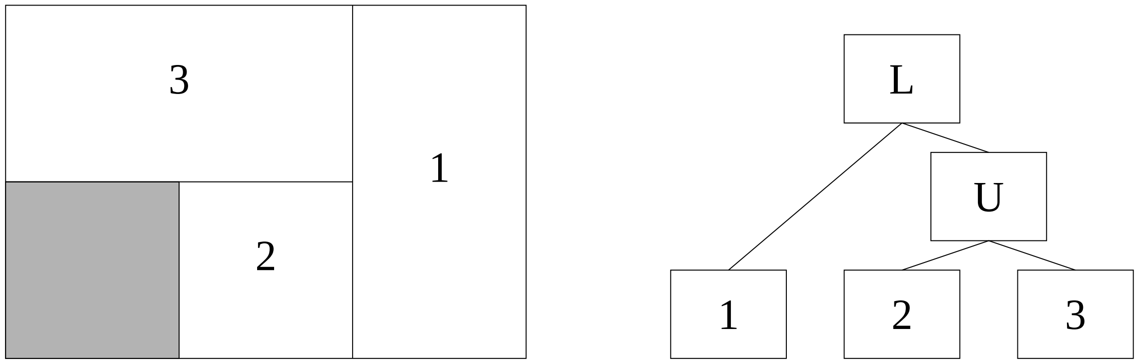
Apart from these observations, I established a set of standard non-identical FLPs and left a benchmark record of many types of GAs. Hence, I believe this work will be a good reference for further research.



(a) a layout including typical non-rectangular regions



(b) a layout without typical non-rectangular regions



(c) a layout without typical non-rectangular regions

Figure 7.2. Adaptive interpretations for STS operators

Bibliography

[Agr93] P. K. Agrawal. Minimising trim loss in cutting rectangular blanks of a single size from a rectangular sheet using orthogonal guillotine cuts. *European Journal of Operational Research*, 64:410–422, 1993.

[Bak85] J. E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 101–111, 1985.

[BAV64] E. S. Buffa, G. C. Armour, and T. E. Vollmann. Allocating facilities with craft. *Harvard Business Review*, 2:136–159, 1964.

[BHS91] T. Baeck, F. Hoffmeister, and H.-P. Schwefel. A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9, 1991.

[BK83] M. S. Bazaraa and O. Kirca. A branch-and-bound-based heuristic for solving the quadratic assignment problem. *Naval Research Logistics Quarterly*, 30:287–304, 1983.

[BMMK92] P. Banerjee, B. Montreuil, C. L. Moodie, and R. L. Kashyap. A modelling of interactive facilities layout designer reasoning using qualitative patterns. *International Journal of Production Research*, 30(3):433–453, 1992.

[Bri81] A. Brindle. Genetic algorithms for function optimization. Technical Report TR81-02, Dept. of Computer Science, University of Alberta, Edmonton, Alberta, Canada, T6G 2H1, 1981.

[BS78] R. E. Burkard and K. H. Stratmann. Numerical investigations on quadratic assignment problems. *Naval Research Logistics Quarterly*, 25:129–148, 1978.

[CD85] H. Carpenter and W. Dowsland. Practical considerations of the pallet loading problem. *Journal of the Operational Research Society*, 36:489–497, 1985.

[CHMR91] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. S. Richards. Distributed genetic algorithms for the floorplan design problem. *IEEE Transactions on Computer-Aided Design*, 10(4):483–492, 1991.

[Chu89] C. H. Chu. Cluster analysis in manufacturing cellular formation. *OMEGA*, 17:289–295, 1989.

[CP87] J. P. Cohoon and W. D. Paris. Genetic placement. *IEEE Transactions on Computer-Aided Design*, 6:956–964, 1987.

[Cul92] J. C. Culberson. Giga program description and operation. Technical Report TR92-06, Dept. of Computer Science, University of Alberta, Edmonton, Alberta, Canada, T6G 2H1, 1992.

[Dav91] L. Davis. *Handbook of Genetic Algorithms*, chapter 2, pages 23–42. Van Nostrand Reinhold, 1991.

[DK85] A. E. Dunlop and B. W. Kernighan. A procedure for placement of standard-cell vlsi circuits. *IEEE Transactions on Computer-Aided Design*, CAD-4(1):92–98, 1985.

[DS82] K. N. Dutta and S. Sahu. A multigoal heuristic for facilities design problem: Mughal. *International Journal of Production Research*, 20:147–154, 1982.

[EGH70] H. K. Edwards, B. E. Gillett, and M. E. Hale. Modular allocation technique (mat). *Management Science*, 17(3):161–169, 1970.

[Egl90] R. W. Eglese. Simulated annealing: a tool for operational research. *European Journal of Operational Research*, 46:271–281, 1990.

[Fal94] E. Falkenauer. Setting new limits in bin packing with a grouping ga using reduction. Technical report, CRIF Technical Report RO108, 1994.

[Fou83] L. R. Foulds. Techniques for facilities layout: deciding which pairs of activities should be adjacent. *Management Science*, 29(12):1414–1426, 1983.

[FRC93] H.-L. Fang, P. Ross, and D. Corne. A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 375–382, 1993.

[GG89] F. Glover and H. J. Greenberg. New approaches for heuristic search: A bilateral linkage with artificial intelligence. *European Journal of Operational Research*, 39:119–130, 1989.

[Gol89] D. E. Goldberg. *Genetic algorithms in search, optimisation, and machine learning.* Addison-Wesley, 1989.

[GP66] J. Gavett and N. Plyter. The optimal assignment of facilities to locations by branch and bound. *Operations Research*, 14:210–232, 1966.

[GW70] G. W. Graves and A. B. Whinston. An algorithm for the quadratic assignment problem. *Management Science*, 17:453–471, 1970.

[HC66] F. Hiller and M. Connors. Quadratic assignment problem algorithms and the location of indivisible facilities. *Management Science*, 13:42–57, 1966.

[HK72] M. Hanan and J. M. Kurtzberg. Placement techniques. In M. A. Breuer, editor, *Design Automation of Digital Systems volume one Theory and Techniques*, chapter 5, pages 213–282. Prentice-Hall, 1972.

[HN93] J. Horn and N. Nafpliotis. Multiobjective optimization using the niched pareto genetic algorithm. Technical Report No. 93005, Illinois Genetic Algorithms Laboratory (IlliGAL), Dept. of General Engineering, Illinois Univ. at Urbana-Champaign, 1993.

[IM89] M. H. Imam and M. Mir. Nonlinear programming approach to automated topology optimization. *Computer-Aided Design*, 21(2):107–115, 1989.

[Joh82] R. V. Johnson. Spacecraft for multi-floor layout planning. *Management Science*, 28:407–417, 1982.

[KB57] T. C. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.

[KB87] R. Kling and P. Banerjee. Esp: A new standard cell placement package using simulated evolution. In *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pages 60–66, 1987.

[KH87] A. Kusiak and S. S. Heragu. The facility layout problem. *European Journal of Operational Research*, 29:229–251, 1987.

[KJK91] Y. Kim, Y. Jang, and M. Kim. Stepwise-overlapped parallel annealing and its application to floorplan designs. *Computer-Aided Design*, 23(2):133–144, 1991.

[Koz92] J.R. Koza. *Genetic Programming on the programming of computers by means of natural selection.* The MIT Press, 1992.

[KS94] D. Kloske and R. Smith. Bulk cable routing using genetic algorithms. Technical Report TCGA Report No. 94001, The Clearinghouse for Genetic Algorithms, The University of Alabama, Dept. of Engineering Science and Mechanics, 1994.

[Lan63] A. H. Land. A problem of assignment with inter-related costs. *Operational Research Quarterly*, 14:185–199, 1963.

[Leu92] J. Leung. A new graph-theoretic heuristic for facility layout. *Management Science*, 38(4):594–605, 1992.

[LGW92] L. L. Larmore, D. D. Gajski, and A. C.-H. Wu. Layout placement for sliced architecture. *IEEE Transactions on Computer-Aided Design*, 11(1):102–114, 1992.

[LM81] R. S. Liggett and W. J. Mitchell. Optimal space planning in practice. *Computer Aided Design*, 13(5):277–288, 1981.

[MV91] B. Montreuil and U. Venkatadri. Strategic interpolative design of dynamic manufacturing systems layout. *Management Science*, 37(6):682–694, 1991.

[Neg74] F. Neghabat. An efficient equipment layout algorithm. *Operations Research*, 22:622–628, 1974.

[NVR68] C. E. Nugent, T. E. Vollmann, and J. Ruml. An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, 16:150–173, 1968.

[Ott82] R. H. J. M. Otten. Automatic floorplan design. In *Proceedings of 19th ACM-IEEE Design Automation Conference*, pages 261–267, 1982.

[RCF94] P. Ross, D. Corne, and H.-L. Fang. Successful lecture timetabling with evolutionary algorithms. In *ECAI '94 Workshop W17: Applied Genetic and other Evolutionary Algorithms*, 1994.

[Rei93] M. P. Reinders. Tactical planning for a cutting stock system. *Journal of the Operational Research Society*, 44(7):645–657, 1993.

[RH95] P. Ross and J. Hallam. Connectionist computing (dai teaching report no.21). Technical report, The Department of Artificial Intelligence, Edinburgh University, 1995.

[Ros79] M. J. Rosenblatt. The facilities layout problem: A multigoal approach. *International Journal of Production Research*, 17:323–332, 1979.

[Ros86] M. J. Rosenblatt. The dynamics of plant layout. *Management Science*, 32:76–86, 1986.

[RR91] A. D. Raoot and A. Rakshit. A 'fuzzy' approach to facilities lay-out planning. *International Journal of Production Research*, 29(4):835–857, 1991.

[SCED89] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 51–60, 1989.

[SDC80] A. Smith and P. De Cani. An algorithm to optimise the layout of boxes in pallets. *Journal of the Operational Research Society*, 31:573–578, 1980.

[SG76] S. Sahni and T. Gonzalez. P-complete approximation problem. *Journal of ACM*, 23:555–565, 1976.

[SLMK92] S. C. Sarin, P. Loharjun, C. J. Malmborg, and B. Krishnakumar. A multiattribute decision-theoretic approach for the layout design problem. *European Journal of Operational Research*, 57:231–242, 1992.

[Smi85] D. Smith. Bin packing with adaptive search. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 202–207, 1985.

[Sou93] A. Souilah. Simulated annealing for manufacturing systems layout design. Technical Report INRIA Reports No. 1909, Institut National de Recherche en Informatique et en Automatique (INRIA), 1993.

[SR91] Y. G. Saab and V. B. Rao. Combinatorial optimization by stochastic evolution. *IEEE Transactions on Computer-Aided Design*, 10(4):525–535, 1991.

[ST93] A. Smith and D. Tate. Genetic optimization using a penalty function. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 499–505, 1993.

[Ste61] L. Steinberg. The backboard wiring problem: A placement algorithm. *SIAM Review*, 3(1):37–50, 1961.

[Ste87] P. Steadman. Architecture - spatial layout. In J. Rooney and P. Steadman, editors, *Principles of Computer-aided Design*, chapter 13, pages 245–262. Pitman, 1987.

[SV85] M. Scriabin and R. C. Vergin. A cluster-analytic approach to facility layout. *Management Science*, 31(1):33–49, 1985.

[Tam92a] K. Y. Tam. Genetic algorithms, function optimization, and facility layout design. *European Journal of Operational Research*, 63(2):322–346, 1992.

[Tam92b] K. Y. Tam. A simulated annealing algorithm for allocating space to manufacturing cells. *International Journal of Production Research*, 30(1):63–87, 1992.

[TL91] K. Y. Tam and S. G. Li. A hierarchical approach to the facility layout problem. *International Journal of Production Research*, 29(1):165–184, 1991.

[Urb92] T. L. Urban. Computational performance and efficiency of lower-bound procedures for the dynamic facility layout problem. *European Journal of Operational Research*, 57:271–279, 1992.

[VCCV91] D. J. Van Camp, M. W. Carter, and A. Vannelli. A nonlinear optimization approach for solving facility layout problems. *European Journal of Operational Research*, 57:174–189, 1991.

[VLW92] J. H. Van Lint and R. M. Wilson. *A Course in Combinatorics*, chapter 14, pages 109–131. Cambridge University Press, 1992.

[WEC91] J. Welkowitz, R. B. Ewen, and J. Cohen. *Introductory Statistics for the Behavioral Sciences*, chapter 8-11, 14-16, pages 93–174, 223–286. Harcourt Brace Jovanovich, 4th edition, 1991.

[Whi89] D. Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121, 1989.

[Whi93] D. Whitley. A genetic algorithm tutorial. Technical Report CS-93-103, Department of Computer Science, Colorado State University, 1993.

[WL86] D. F. Wong and C. L. Liu. A new algorithm for floorplan design. In *Proceedings of 23rd ACM/IEEE Design Automation Conference*, pages 101–107, 1986.

[WO94] M. Wineberg and F. Oppacher. A canonical genetic algorithm based approach to genetic programming. In *ECAI '94 Workshop W17: Applied Genetic and other Evolutionary Algorithms*, 1994.

[YP93] P. Yip and Y. Pao. A parallel and distributed processing algorithm for facility layout. In J. Wang and Y. Takefuji, editors, *NEURAL NETWORKS IN DESIGN AND MANUFACTURING*, chapter 3, pages 77–97. World Scientific, 1993.

[YZH91] H. H. Yanasse, A. S. I. Zinober, and R. G. Harris. Two-dimensional cutting stock with multiple stock sizes. *Journal of the Operational Research Society*, 42(8):673–683, 1991.

Appendix A

Specifications of Fifteen Standard FLPs

Keys:

`@number`	=	the number of facilities
`@traffic`	=	the matrix of traffic frequency between facilities
`@area`	=	minimum area required by each facility
`@aspect`	=	the aspect ratio limitations of each facility and the orientation limitation (`free` or `rigid`)
`@distance_measure`	=	the method to measure the distance of two facilities (`manhattan` or `euclidian`)
`@eval_method`	=	the notation† of evaluation function
`@room`	=	the room area denoted by (`0 0` *width length*)
`@objects` (first line)	=	the number of prespecified areas
`@objects` (other lines)	=	the position of each prespecified area denoted by $(X_{left}, Y_{bottom}, X_{right}, Y_{top})$

†notation	a	b	Pa	Pb	Pc
`TxDx2plusAREA`	1	1	2	1000000	1
`TxDxDdiv2`	1	2	0.5	1000000	1
`TxDx2plus100xASP_ratio`	1	1	2	100	0
`TxD`	1	1	1	1000000	0

The values a, b, Pa, Pb, Pc correspond to Formula (3.1).

```
# Kea91-11

@number
11

@traffic
  0  2  1  1  2  6  2  6  6  3  6
  2  0  1  1  2  6  4  6  6  3  6
  1  1  0  2  2  6  1  6  6  6  6
  1  1  2  0  1  5  1  6  6  3  6
  2  2  2  1  0  4  3  6  4  5  6
  6  6  6  5  4  0  3  6  4  5  6
  2  4  1  1  3  3  0  4  4  1  1
  6  6  6  6  6  6  4  0  6  3  3
  6  6  6  6  4  4  4  6  0  5  5
  3  3  6  3  5  5  1  3  5  0  2
  6  6  6  6  6  6  1  3  5  2  0

@area
16
2
2
10
6
7
12
5.2
11.2
28
25

@aspect
1.0     1.0     rigid
2.0     2.0     rigid
2.0     2.0     rigid
2.5     2.5     rigid
0.666   0.666   rigid
3.571   3.571   rigid
0.75    0.75    rigid
0.769   0.769   rigid
0.7     0.7     rigid
1.75    1.75    rigid
1.0     1.0     rigid

@distance_measure
manhattan

@eval_method
TxDx2plusAREA
```

Figure A.1. FLP specifications of Kea91-11

```
# Kea91-11a (free aspect ratio [0.25, 4.0])

@number
11

@traffic
  0  2  1  1  2  6  2  6  6  3  6
  2  0  1  1  2  6  4  6  6  3  6
  1  1  0  2  2  6  1  6  6  6  6
  1  1  2  0  1  5  1  6  6  3  6
  2  2  2  1  0  4  3  6  4  5  6
  6  6  6  5  4  0  3  6  4  5  6
  2  4  1  1  3  3  0  4  4  1  1
  6  6  6  6  6  6  4  0  6  3  3
  6  6  6  6  4  4  4  6  0  5  5
  3  3  6  3  5  5  1  3  5  0  2
  6  6  6  6  6  6  1  3  5  2  0

@area
16
2
2
10
6
7
12
5.2
11.2
28
25

@aspect
0.25    4.0     rigid
0.25    4.0     rigid
0.25    4.0     rigid
0.25    4.0     rigid
0.25    4.0     rigid
0.25    4.0     rigid
0.25    4.0     rigid
0.25    4.0     rigid
0.25    4.0     rigid
0.25    4.0     rigid
0.25    4.0     rigid

@distance_measure
manhattan

@eval_method
TxDx2plusAREA
```

Figure A.2. FLP specifications of Kea91-11a

```
# Kea91-16a

@number
16

@traffic
  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0
  1  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0
  0  1  0  1  0  0  1  0  0  0  0  0  0  0  0  0
  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0
  1  0  0  0  0  1  0  0  1  0  0  0  0  0  0  0
  0  1  0  0  1  0  1  0  0  1  0  0  0  0  0  0
  0  0  1  0  0  1  0  1  0  0  1  0  0  0  0  0
  0  0  0  1  0  0  1  0  0  0  0  1  0  0  0  0
  0  0  0  0  1  0  0  0  0  1  0  0  1  0  0  0
  0  0  0  0  0  1  0  0  1  0  1  0  0  1  0  0
  0  0  0  0  0  0  1  0  0  1  0  1  0  0  1  0
  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  1
  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0
  0  0  0  0  0  0  0  0  0  1  0  0  1  0  1  0
  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  1
  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0

@area
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

@aspect
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid
1.0     1.0     rigid

@distance_measure
manhattan

@eval_method
TxDx2plusAREA
```

Figure A.3. FLP specifications of Kea91-16

```
# Kea91-20a

@number
20

@traffic
  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  1  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0
  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  1  0  0  1  0  1  0  1  0  0  0  1  0  0  0  0  0  0
  0  0  0  1  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0
  0  0  0  1  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  1  1  0  0  1  0  0  0  1  0  0  0  0  0
  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0
  0  0  0  0  1  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0
  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  1  0  1  0
  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  1  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  1
  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0

@area
1 #facility 1
1 #facility 2
1 #facility 3
4 #facility 4
2 #facility 5
2 #facility 6
3 #facility 7
1 #facility 8
3 #facility 9
2 #facility 10
1 #facility 11
2 #facility 12
2 #facility 13
2 #facility 14
1 #facility 15
2 #facility 16
5 #facility 17
1 #facility 18
1 #facility 19
5 #facility 20

@aspect
1       1       free #facility 1
1       1       free #facility 2
1       1       free #facility 3
4       4       free #facility 4
2       2       free #facility 5
2       2       free #facility 6
3       3       free #facility 7
1       1       free #facility 8
3       3       free #facility 9
2       2       free #facility 10
1       1       free #facility 11
2       2       free #facility 12
2       2       free #facility 13
2       2       free #facility 14
1       1       free #facility 15
2       2       free #facility 16
5       5       free #facility 17
1       1       free #facility 18
1       1       free #facility 19
5       5       free #facility 20

@distance_measure
manhattan

@eval_method
TxDx2plusAREA
```

Figure A.4. FLP specifications of Kea91-20

```
# TL91-5

@number
5

@traffic
  0  5  2  4  1
  5  0  3  0  2
  2  3  0  0  0
  4  0  0  0  5
  1  2  0  5  0

@area
24
16
36
8
21

@aspect
0.8     1       free
0.75    1.15    free
0.6     1.85    free
0.3     1.1     free
0.9     1.18    free

@distance_measure
euclidian

@eval_method
TxDxDdiv2
```

```
# TL91-6

@number
6

@traffic
  0  5  2  4  1  0
  5  0  3  0  2  2
  2  3  0  0  0  0
  4  0  0  0  5  2
  1  2  0  5  0 10
  0  2  0  2 10  0

@area
24
16
36
8
21
17.5

@aspect
0.8     1       free
0.75    1.15    free
0.6     1.85    free
0.3     1.1     free
0.9     1.18    free
0.5     1       free

@distance_measure
euclidian

@eval_method
TxDxDdiv2
```

```
# TL91-7

@number
7

@traffic
  0  5  2  4  1  0  0
  5  0  3  0  2  2  2
  2  3  0  1  0  2  5
  4  0  1  0  5  2  2
  1  2  0  5  0 10  0
  0  2  2  2 10  0  5
  0  2  5  2  0  5  0

@area
24
16
36
8
21
17.5
3.6

@aspect
0.8     1       free
0.75    1.15    free
0.6     1.85    free
0.3     1.1     free
0.9     1.18    free
0.5     1       free
0.3     1.4     free

@distance_measure
euclidian

@eval_method
TxDxDdiv2
```

```
# TL91-8

@number
8

@traffic
  0  5  2  4  1  0  0  6
  5  0  3  0  2  2  2  0
  2  3  0  0  0  0  0  5
  4  0  0  0  5  2  2 10
  1  2  0  5  0 10  0  0
  0  2  0  2 10  0  5  1
  0  2  0  2  0  5  0 10
  6  0  5 10  0  1 10  0

@area
24
16
36
8
21
17.5
3.6
15.4

@aspect
0.8     1       free
0.75    1.15    free
0.6     1.85    free
0.3     1.1     free
0.9     1.18    free
0.5     1       free
0.3     1.4     free
0.6     1.25    free

@distance_measure
euclidian

@eval_method
TxDxDdiv2
```

Figure A.5. FLP specifications of TL91-5,6,7,8

```
# TL91-12

@number
12

@traffic
   0   5   2   4   1   0   0   6   2   1   1   1
   5   0   3   0   2   2   2   0   4   5   0   0
   2   3   0   0   0   0   0   5   5   2   2   2
   4   0   0   0   5   2   2  10   0   0   5   5
   1   2   0   5   0  10   0   0   0   5   1   1
   0   2   0   2  10   0   5   1   1   5   4   0
   0   2   0   2   0   5   0  10   5   2   3   3
   6   0   5  10   0   1  10   0   0   0   5   0
   2   4   5   0   0   1   5   0   0   0  10  10
   1   5   2   0   5   5   2   0   0   0   5   0
   1   0   2   5   1   4   3   5  10   5   0   2
   1   0   2   5   1   0   3   0  10   0   2   0

@area
24
16
36
8
21
17.5
3.6
15.4
20
19.5
16
9

@aspect
0.8     1       free
0.75    1.15    free
0.6     1.85    free
0.3     1.1     free
0.9     1.18    free
0.5     1       free
0.3     1.4     free
0.6     1.25    free
0.9     1       free
1.2     1.8     free
0.85    1.1     free
0.9     1       free

@distance_measure
euclidian

@eval_method
TxDxDdiv2
```

Figure A.6. FLP specifications of TL91-12

```
# TL91-15

@number
15

@traffic
  0 10  0  5  1  0  1  2  2  2  2  0  4  0  0
 10  0  1  3  2  2  2  3  2  0  2  0 10  5  0
  0  1  0 10  2  0  2  5  4  5  2  2  5  5  5
  5  3 10  0  1  1  5  0  0  2  1  0  2  5  0
  1  2  2  1  0  3  5  5  5  1  0  3  0  5  5
  0  2  0  1  3  0  2  2  1  5  0  0  2  5 10
  1  2  2  5  5  2  0  6  0  1  5  5  5  1  0
  2  3  5  0  5  2  6  0  5  2 10  0  5  0  0
  2  2  4  0  5  1  0  5  0  0 10  5 10  0  2
  2  0  5  2  1  5  1  2  0  0  0  4  0  0  5
  2  2  2  1  0  0  5 10 10  0  0  5  0  5  0
  0  0  2  0  3  0  5  0  5  4  5  0  3  3  0
  4 10  5  2  0  2  5  5 10  0  0  3  0 10  2
  0  5  5  5  5  5  1  0  0  0  5  3 10  0  4
  0  0  5  0  5 10  0  0  2  5  0  0  2  4  0

@area
24
16
36
8
21
17.5
3.6
15.4
20
19.5
16
9
9
25
4

@aspect
0.8     1       free
0.75    1.15    free
0.6     1.85    free
0.3     1.1     free
0.9     1.18    free
0.5     1       free
0.3     1.4     free
0.6     1.25    free
0.9     1       free
1.2     1.8     free
0.85    1.1     free
0.9     1       free
0.8     1.1     free
0.92    1.05    free
0.85    1.15    free

@distance_measure
euclidian

@eval_method
TxDxDdiv2
```

Figure A.7. FLP specifications of TL91-15

```
# TL91-20

@number
20

@traffic
   0   0   5   0   5   2  10   3   1   5   5   5   0   0   5   4   4   0   0   1
   0   0   3  10   5   1   5   1   2   4   2   5   0  10  10   3   0   5  10   5
   5   3   0   2   0   5   2   4   4   5   0   0   0   5   1   0   0   5   0   0
   0  10   2   0   1   0   5   2   1   0  10   2   2   0   2   1   5   2   5   5
   5   5   0   1   0   5   6   5   2   5   2   0   5   1   1   1   5   2   5   1
   2   1   5   0   5   0   5   2   1   6   0   0  10   0   2   0   1   0   1   5
  10   5   2   5   6   5   0   0   0   0   5  10   2   2   5   1   2   1   0  10
   3   1   4   2   5   2   0   0   1   1  10  10   2   0  10   2   5   2   2  10
   1   2   4   1   2   1   0   1   0   2   0   3   5   5   0   5   0   0   0   2
   5   4   5   0   5   6   0   1   2   0   5   5   0   5   1   0   0   5   5   2
   5   2   0  10   2   0   5  10   0   5   0   5   2   5   1  10   0   2   2   5
   5   5   0   2   0   0  10  10   3   5   5   0   2  10   5   0   1   1   2   5
   0   0   0   2   5  10   2   2   5   0   2   2   0   2   2   1   0   0   0   5
   0  10   5   0   1   0   2   0   5   5   5  10   2   0   5   5   1   5   5   0
   5  10   1   2   1   2   5  10   0   1   1   5   2   5   0   3   0   5  10  10
   4   3   0   1   1   0   1   2   5   0  10   0   1   5   3   0   0   0   2   0
   4   0   0   5   5   1   2   5   0   0   0   1   0   1   0   0   0   5   2   0
   0   5   5   2   2   0   1   2   0   5   2   1   0   5   5   0   5   0   1   1
   0  10   0   5   5   1   0   2   0   5   2   2   0   5  10   2   2   1   0   6
   1   5   0   5   1   5  10  10   2   2   5   5   5   0  10   0   0   1   6   0

@area
24
16
36
8
21
17.5
3.6
15.4
20
19.5
16
9
9
25
4
3
4
9
4.5
5

@aspect
0.8     1       free
0.75    1.15    free
0.6     1.85    free
0.3     1.1     free
0.9     1.18    free
0.5     1       free
0.3     1.4     free
0.6     1.25    free
0.9     1       free
1.2     1.8     free
0.85    1.1     free
0.9     1       free
0.8     1.1     free
0.92    1.05    free
0.85    1.15    free
0.9     1       free
1       1       free
1       1       free
0.7     1.1     free
0.5     1.5     free

@distance_measure
euclidian

@eval_method
TxDxDdiv2
```

Figure A.8. FLP specifications of TL91-20

```
# TL91-30

@number
30

@traffic
   0  3  2  0  0  2 10  5  0  5  2  5  0  0  2  0  5  6  3  0  1 10  0 10  2  1  1  1  0  1
   3  0  4  0 10  4  0  0  2  2  1  0  5  0  0  0  0  2  0  1  6  1  0  1  2  2  5  1 10  5
   2  4  0  3  4  0  5  5  5  1  4  1  0  4  0  4  0  6  3  2  5  5  2  1  0  0  3  1  0  2
   0  0  3  0  0  0  0  2  2  0  6  0  2  5  2  5  1  1  1  1  2  2  4  0  2  0  2  2  5  5
   0 10  4  0  0  5  2  0  0  0  0  2  0  0  0  0  2  1  0  0  2  0  5  1  0  2  1  0  2  1
   2  4  0  0  5  0  1  2  2  1  4 10 10  2  5  5  0  5  0  0  0 10  0  0  0  4  0 10  1  1
  10  0  5  0  2  1  0 10 10  5 10 10  6  0  0 10  2  1 10  1  5  5  2  3  5  0  2  0  1  3
   5  0  5  2  0  2 10  0  1  3  5  0  0  0  2  4  5  2 10  6  0  5  5  2  5  0  5  5  0  2
   0  2  5  2  0  2 10  1  0 10  2  1  5  2  0  3  0  2  0  0  4  0  5  2  0  5  2  2  5  2
   5  2  1  0  0  1  5  3 10  0  5  5  6  0  1  5  5  0  5  2  3  5  0  5  2 10 10  1  5  2
   2  1  4  6  0  4 10  5  2  5  0  0  0  1  2  1  0  2  0  0  0  6  6  0  4  5  3  2  2 10
   5  0  1  0  2 10 10  0  1  5  0  0  5  5  2  0  0  0  0  2  0  4  5 10  1  0  0  0  0  1
   0  5  0  2  0 10  6  0  5  6  0  5  0  2  0  4  2  2  1  0  6  2  1  5  5  0  0  1  5  5
   0  0  4  5  0  2  0  0  2  0  1  5  2  0  2  1  0  5  3 10  0  0  4  2  0  0  4  2  5  5
   2  0  0  2  0  5  0  2  0  1  2  2  0  2  0  4  5  1  0  1  0  5  0  2  0  0  5  1  1  0
   0  0  4  5  0  5 10  4  3  5  1  0  4  1  4  0  0  3  0  2  2  0  2  0  5  0  5  2  5 10
   5  0  0  1  2  0  2  5  0  5  0  0  2  0  5  0  0  2  2  0  0  0  6  5  3  5  0  0  5  1
   6  2  6  1  1  5  1  2  2  0  2  0  2  5  1  3  2  0  5  1  2 10 10  4  0  0  5  0  0  0
   3  0  3  1  0  0 10 10  0  5  0  0  1  3  0  0  2  5  0  0  5  5  1  0  5  2  1  2 10 10
   0  1  2  1  0  0  1  6  0  2  0  2  0 10  1  2  0  1  0  0  5  2  1  3  1  5  6  5  5  3
   1  6  5  2  2  0  5  0  4  3  0  0  6  0  0  2  0  2  5  5  0  4  0  1  0  0  0  5  0  0
  10  1  5  2  0 10  5  5  0  5  6  4  2  0  5  0  0 10  5  2  4  0  5  0  4  4  5  0  2  5
   0  0  2  4  5  0  2  5  5  0  6  5  1  4  0  2  6 10  1  1  0  5  0  0  4  4  1  0  2  2
  10  1  1  0  1  0  3  2  2  5  0 10  5  2  2  0  5  4  0  3  1  0  0  0  5  5  0  1  0  0
   2  2  0  2  0  0  5  5  0  2  4  1  5  0  0  5  3  0  5  1  0  4  4  5  0  1  0 10  1  0
   1  2  0  0  2  4  0  0  5 10  5  0  0  0  0  0  5  0  2  5  0  4  4  5  1  0  0  0  0  0
   1  5  3  2  1  0  2  5  2 10  3  0  0  4  5  5  0  5  1  6  0  5  1  0  0  0  0  0  0 10
   1  1  1  2  0 10  0  5  2  1  2  0  1  2  1  2  0  0  2  5  5  0  0  1 10  0  0  0  2  2
   0 10  0  5  2  1  1  0  5  5  2  0  5  5  1  5  5  0 10  5  0  2  2  0  1  0  0  2  0  2
   1  5  2  5  1  1  3  2  2  2 10  1  5  5  0 10  1  0 10  3  0  5  2  0  0  0 10  2  2  0

@area
24
16
36
8
21
17.5
3.6
15.4
20
19.5
16
9
9
25
4
3
4
9
4.5
5
16
9
9
25
4
3
4
9
4.5
5

@aspect
0.8     1       free
0.75    1.15    free
0.6     1.85    free
0.3     1.1     free
0.9     1.18    free
0.5     1       free
0.3     1.4     free
0.6     1.25    free
0.9     1       free
1.2     1.8     free
0.85    1.1     free
0.9     1       free
0.8     1.1     free
0.92    1.05    free
0.85    1.15    free
0.9     1       free
1       1       free
1       1       free
0.7     1.1     free
0.5     1.5     free
0.85    1.1     free
0.9     1       free
0.8     1.1     free
0.92    1.05    free
0.85    1.15    free
0.9     1.0     free
1       1       free
1       1       free
0.7     1.1     free
0.5     1.5     free

@distance_measure
euclidian

@eval_method
TxDxDdiv2
```

Figure A.9. FLP specifications of TL91-30

```
# Tam92-20a

@number
20

@traffic
   0   0   5   0   5   2  10   3   1   5   5   5   0   0   5   4   4   0   0   1
   0   0   3  10   5   1   5   1   2   4   2   5   0  10  10   3   0   5  10   5
   5   3   0   2   0   5   2   4   4   5   0   0   0   5   1   0   0   5   0   0
   0  10   2   0   1   0   5   2   1   0  10   2   2   0   2   1   5   2   5   5
   5   5   0   1   0   5   6   5   2   5   2   0   5   1   1   1   5   2   5   1
   2   1   5   0   5   0   5   2   1   6   0   0  10   0   2   0   1   0   1   5
  10   5   2   5   6   5   0   0   0   0   5  10   2   2   5   1   2   1   0  10
   3   1   4   2   5   2   0   0   1   1  10  10   2   0  10   2   5   2   2  10
   1   2   4   1   2   1   0   1   0   2   0   3   5   5   0   5   0   0   0   2
   5   4   5   0   5   6   0   1   2   0   5   5   0   5   1   0   0   5   5   2
   5   2   0  10   2   0   5  10   0   5   0   5   2   5   1  10   0   2   2   5
   5   5   0   2   0   0  10  10   3   5   5   0   2  10   5   0   1   1   2   5
   0   0   0   2   5  10   2   2   5   0   2   2   0   2   2   1   0   0   0   5
   0  10   5   0   1   0   2   0   5   5   5  10   2   0   5   5   1   5   5   0
   5  10   1   2   1   2   5  10   0   1   1   5   2   5   0   3   0   5  10  10
   4   3   0   1   1   0   1   2   5   0  10   0   1   5   3   0   0   0   2   0
   4   0   0   5   5   1   2   5   0   0   0   1   0   1   0   0   0   5   2   0
   0   5   5   2   2   0   1   2   0   5   2   1   0   5   5   0   5   0   1   1
   0  10   0   5   5   1   0   2   0   5   2   2   0   5  10   2   2   1   0   6
   1   5   0   5   1   5  10  10   2   2   5   5   5   0  10   0   0   1   6   0

@area
100
80
50
60
120
40
20
40
150
120
50
10
20
30
50
20
40
20
80
100

@aspect
0.7     1       free
1       1       free
0.7     1.3     free
0.5     0.8     free
0.9     1       free
0.6     1       free
0.7     1.4     free
1       1       free
0.8     1.1     free
0.5     1.5     free
0.7     1.1     free
0.8     1.2     free
0.95    1.5     free
0.75    1.25    free
0.9     1.1     free
0.8     1.5     free
0.4     1.4     free
0.9     1.9     free
1       1       free
0.95    1.15    free

@distance_measure
manhattan

@eval_method
TxDx2plus100xASP_ratio

@room
0 0 40 35
@objects
4
0 0 10 5
30 30 40 35
0 30 10 35
20 20 30 25
```

Figure A.10. FLP specifications of Tam92-20a

```
# Tam92-30a

@number
30

@traffic
   0   3   2   0   0   2  10   5   0   5   2   5   0   0   2   0   5   6   3   0   1  10   0  10   2   1   1   1   0   1
   3   0   4   0  10   4   0   0   2   2   1   0   5   0   0   0   0   2   0   1   6   1   0   1   2   2   5   1  10   5
   2   4   0   3   4   0   5   5   5   1   4   1   0   4   0   4   0   6   3   2   5   5   2   1   0   0   3   1   0   2
   0   0   3   0   0   0   0   2   2   0   6   0   2   5   2   5   1   1   1   1   2   2   4   0   2   0   2   2   5   5
   0  10   4   0   0   5   2   0   0   0   0   2   0   0   0   0   2   1   0   0   2   0   5   1   0   2   1   0   2   1
   2   4   0   0   5   0   1   2   2   1   4  10  10   2   5   5   0   5   0   0   0  10   0   0   0   4   0  10   1   1
  10   0   5   0   2   1   0  10  10   5  10  10   6   0   0  10   2   1  10   1   5   5   2   3   5   0   2   0   1   3
   5   0   5   2   0   2  10   0   1   3   5   0   0   0   2   4   5   2  10   6   0   5   5   2   5   0   5   5   0   2
   0   2   5   2   0   2  10   1   0  10   2   1   5   2   0   3   0   2   0   0   4   0   5   2   0   5   2   2   5   2
   5   2   1   0   0   1   5   3  10   0   5   5   6   0   1   5   5   0   5   2   3   5   0   5   2  10  10   1   5   2
   2   1   4   6   0   4  10   5   2   5   0   0   0   1   2   1   0   2   0   0   0   6   6   0   4   5   3   2   2  10
   5   0   1   0   2  10  10   0   1   5   0   0   5   5   2   0   0   0   0   2   0   4   5  10   1   0   0   0   0   1
   0   5   0   2   0  10   6   0   5   6   0   5   0   2   0   4   2   2   1   0   6   2   1   5   5   0   0   1   5   5
   0   0   4   5   0   2   0   0   2   0   1   5   2   0   2   1   0   5   3  10   0   0   4   2   0   0   4   2   5   5
   2   0   0   2   0   5   0   2   0   1   2   2   0   2   0   4   5   1   0   1   0   5   0   2   0   0   5   1   1   0
   0   0   4   5   0   5  10   4   3   5   1   0   4   1   4   0   0   3   0   2   2   0   2   0   5   0   5   2   5  10
   5   0   0   1   2   0   2   5   0   5   0   0   2   0   5   0   0   2   2   0   0   0   6   5   3   5   0   0   5   1
   6   2   6   1   1   5   1   2   2   0   2   0   2   5   1   3   2   0   5   1   2  10  10   4   0   0   5   0   0   0
   3   0   3   1   0   0  10  10   0   5   0   0   1   3   0   0   2   5   0   0   5   5   1   0   5   2   1   2  10  10
   0   1   2   1   0   0   1   6   0   2   0   2   0  10   1   2   0   1   0   0   5   2   1   3   1   5   6   5   5   3
   1   6   5   2   2   0   5   0   4   3   0   0   6   0   0   2   0   2   5   5   0   4   0   1   0   0   0   5   0   0
  10   1   5   2   0  10   5   5   0   5   6   4   2   0   5   0   0  10   5   2   4   0   5   0   4   4   5   0   2   5
   0   0   2   4   5   0   2   5   5   0   6   5   1   4   0   2   6  10   1   1   0   5   0   0   4   4   1   0   2   2
  10   1   1   0   1   0   3   2   2   5   0  10   5   2   2   0   5   4   0   3   1   0   0   0   5   5   0   1   0   0
   2   2   0   2   0   0   5   5   0   2   4   1   5   0   0   5   3   0   5   1   0   4   4   5   0   1   0  10   1   0
   1   2   0   0   2   4   0   0   5  10   5   0   0   0   0   0   5   0   2   5   0   4   4   5   1   0   0   0   0   0
   1   5   3   2   1   0   2   5   2  10   3   0   0   4   5   5   0   5   1   6   0   5   1   0   0   0   0   0   0  10
   1   1   1   2   0  10   0   5   2   1   2   0   1   2   1   2   0   0   2   5   5   0   0   1  10   0   0   0   2   2
   0  10   0   5   2   1   1   0   5   5   2   0   5   5   1   5   5   0  10   5   0   2   2   0   1   0   0   2   0   2
   1   5   2   5   1   1   3   2   2   2  10   1   5   5   0  10   1   0  10   3   0   5   2   0   0   0  10   2   2   0

@area
100
80
50
60
120
40
20
40
150
120
50
10
20
30
50
20
40
20
80
100
40
50
80
10
40
10
40
10
80
40

@aspect
0.7     1       free
1       1       free
0.7     1.3     free
0.5     0.8     free
0.9     1       free
0.6     1       free
0.7     1.4     free
1       1       free
0.8     1.1     free
0.5     1.5     free
0.7     1.1     free
0.8     1.2     free
0.95    1.5     free
0.75    1.25    free
0.9     1.1     free
0.8     1.5     free
0.4     1.4     free
0.9     1.9     free
1       1       free
0.95    1.15    free
0.5     1.5     free
1       1.1     free
0.6     1       free
0.9     1       free
0.8     1.1     free
0.5     1.2     free
0.8     1       free
0.5     1.3     free
0.9     1.05    free
0.9     1.1     free

@distance_measure
manhattan

@eval_method
TxDx2plus100xASP_ratio

@room
0 0 45 40
@objects
3
 0  0 10 10
40  0 45 10
35 35 45 40
```

Figure A.11. FLP specifications of Tam92-30a

```
# VCea91-10

@good_cluster
15%0%28%%79%4%36%%%

@number
10

@traffic
   0.0    0.0    0.0    0.0    0.0  218.0    0.0    0.0    0.0    0.0
   0.0    0.0    0.0    0.0    0.0  148.0    0.0    0.0  296.0    0.0
   0.0    0.0    0.0   28.0   70.0    0.0    0.0    0.0    0.0    0.0
   0.0    0.0   28.0    0.0    0.0   28.0   70.0  140.0    0.0    0.0
   0.0    0.0   70.0    0.0    0.0    0.0    0.0  210.0    0.0    0.0
 218.0  148.0    0.0   28.0    0.0    0.0    0.0    0.0    0.0    0.0
   0.0    0.0    0.0   70.0    0.0    0.0    0.0    0.0    0.0   28.0
   0.0    0.0    0.0  140.0  210.0    0.0    0.0    0.0    0.0  888.0
   0.0  296.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0   59.2
   0.0    0.0    0.0    0.0    0.0    0.0   28.0  888.0   59.2    0.0

@area
238
112
160
80
120
80
60
85
221
119

@aspect
0.105   9.52    rigid
0.223   4.48    rigid
0.156   6.40    rigid
0.312   3.20    rigid
0.208   4.80    rigid
0.312   3.20    rigid
0.417   2.40    rigid
0.294   3.40    rigid
0.113   8.84    rigid
0.211   4.76    rigid

@distance_measure
euclidian

@eval_method
TxD

@room
0 0 25 51
```

Figure A.12. FLP specifications of VCea91-10

Appendix B

The Performance of GAs

This appendix shows the performance of each GA in detail. The name of GA is expressed by the following form.

rrraaa/P*ggg*.Pp*ppp*

where *rrr*	=	reproduction method (`gen, one, two`)
aaa	=	algorithm (`Cea, Tam, DK, Tam2, DK2, Kad`)
ggg	=	the number of populations (1, 4, 10)
ppp	=	population size $\times$ the number of populations (50, 200, 1000) or `best` (when the best GA parameters found in the parameter investigation are used.)

Keys:

best_score	the best of best individual scores
mean_score	the mean of best individual scores
std_dev	the standard deviation of best individual scores
sample	the number of experiments
eval	the maximum number of evaluations
diagnosis	the result of comparing performance of each GA with that of original paper's algorithm (`better, -, worse`)†
t-value	the t-value of each GA's performance respect to the original paper's (if t-tests are used for the comparison)

†The comparison method is described in Table 6.17.

```
problem = Kea91-11
                        best_score      mean_score       std_dev     sample    eval            diagnosis       t-value
original_paper          2829.400000     2829.400000      0.000000       1      -

my experiments
   genCea/P1.Pp1000     2852.661133     2947.112305     62.122997      10      200000          -
   genCea/P4.Pp1000     2765.061279     2940.389648     77.200691      10      200000          better
   genCea/P10.Pp1000    2933.001221     3056.546387     81.520012      10      200000          -
   genCea/P1.Pp200      2916.966553     3011.881103     58.604759      10      200000          -
   genCea/P1.Pp50       2994.816406     3057.902344     32.638683      10      200000          -
   genCea/P1.Ppbest     3028.785156     3413.997559    205.475281      10      200000          -
   oneCea/P1.Pp1000     2816.974365     2941.817871    103.407921      10       74000          better
   twoCea/P1.Pp1000     2834.389404     2945.520020    132.576370      10       41000          -

   genTam/P1.Pp1000     3327.913574     3328.109375      0.619740      10      200000          -
   genTam/P4.Pp1000     3327.913574     3328.109375      0.619740      10      200000          -
   genTam/P10.Pp1000    3327.913574     3327.913574      0.000041      10      200000          -
   genTam/P1.Pp200      3327.913574     3342.455322     19.662620      10      200000          -
   genTam/P1.Pp50       3327.913574     3344.480469     21.387920      10      200000          -
   genTam/P1.Ppbest     3327.913574     3327.913574      0.000000      10      200000          -
   oneTam/P1.Pp1000     3327.913574     3327.913574      0.000000      10       20000          -
   twoTam/P1.Pp1000     3327.913574     3327.913574      0.000000      10       10000          -

   genDK/P1.Pp1000      3241.334228     3241.334228      0.000081      10      200000          -
   genDK/P1.Ppbest      3241.334228     3273.850830     54.609757      10      200000          -
   twoDK/P1.Pp1000      3241.334228     3241.334228      0.000081      10        8000          -

   genTam2/P1.Pp1000    2772.280029     2946.642822    127.118462      10      200000          better
   twoTam2/P1.Pp1000    2744.193115     3005.646973    204.446289      10       33000          better

   genDK2/P1.Pp1000     3066.779785     3094.827148     26.400452      10      200000          -
   twoDK2/P1.Pp1000     3100.447754     3176.693848     69.257217      10       26000          -

   genKad/P1.Pp1000     2800.280029     3004.087158     87.055534      10      200000          better
   genKad/P4.Pp1000     2800.280029     3018.826416     99.182755      10      200000          better
   genKad/P10.Pp1000    2966.684082     3074.645264     58.015877      10      200000          -
   genKad/P1.Pp200      2963.902832     3042.961182     70.183937      10      200000          -
   genKad/P1.Pp50       2932.937256     3064.782959     71.453758      10      200000          -
   oneKad/P1.Pp1000     3071.827393     3151.110107     70.071121      10       48000          -
   twoKad/P1.Pp1000     2963.902832     3127.936279     95.128410      10       18000          -
```

Figure B.1: The performance of GAs for Kea91-11

```
problem = Kea91-11a
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper          2287.041000     2287.041000       0.000000       1      -

my experiments
   genCea/P1.Pp1000     2232.043213     2266.870850      23.924580      10      200000          better
   genCea/P4.Pp1000     2213.655518     2277.564209      30.692774      10      200000          better
   genCea/P10.Pp1000    2277.416260     2317.588135      43.922474      10      200000          better
   genCea/P1.Pp200      2206.779053     2243.710205      38.093525      10      200000          better
   genCea/P1.Pp50       2232.043213     2251.189941      12.912111      10      200000          better
   genCea/P1.Ppbest     2180.068603     2264.203857      41.086845      10      200000          better
   oneCea/P1.Pp1000     2267.491943     2340.458252      29.198959      10       50000          better
   twoCea/P1.Pp1000     2260.478027     2329.215820      38.194420      10       30000          better

   genTam/P1.Pp1000     2460.450439     2460.450439       0.000050      10      200000          -
   genTam/P4.Pp1000     2460.450439     2471.583008       5.867328      10      200000          -
   genTam/P10.Pp1000    2460.450439     2463.233643       5.867328      10      200000          -
   genTam/P1.Pp200      2460.450439     2471.652832       5.908186      10      200000          -
   genTam/P1.Pp50       2460.450439     2470.470947       6.921518      10      200000          -
   genTam/P1.Ppbest     2460.450439     2466.086670       7.279009      10      200000          -
   oneTam/P1.Pp1000     2460.450439     2460.450439       0.000050      10       20000          -
   twoTam/P1.Pp1000     2460.450439     2460.450439       0.000050      10       13000          -

   genDK/P1.Pp1000      2758.393555     2758.393555       0.000041      10      200000          -
   genDK/P1.Ppbest      2758.393555     2758.393555       0.000041      10      200000          -
   twoDK/P1.Pp1000      2758.393555     2758.393555       0.000041      10       12000          -

   genTam2/P1.Pp1000    2303.208252     2307.812012       8.573215      10      200000          -
   twoTam2/P1.Pp1000    2303.208252     2342.958496      48.601520      10       22000          -

   genDK2/P1.Pp1000     2245.970703     2328.799561     132.111588      10      200000          better
   twoDK2/P1.Pp1000     2258.293701     2492.931152     223.174469      10       29000          better

   genKad/P1.Pp1000     2303.208252     2308.231689       8.492771      10      200000          -
   genKad/P4.Pp1000     2303.208252     2313.260986      10.714998      10      200000          -
   genKad/P10.Pp1000    2303.932373     2326.137207      11.109127      10      200000          -
   genKad/P1.Pp200      2276.579346     2307.566650      24.605682      10      200000          better
   genKad/P1.Pp50       2231.923584     2299.942871      25.222481      10      200000          better
   oneKad/P1.Pp1000     2303.208252     2329.415283      47.102593      10       38000          -
   twoKad/P1.Pp1000     2303.208252     2318.586914      15.608057      10       24000          -
```

Figure B.2: The performance of GAs for Kea91-11a

```
problem = Kea91-16
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper           64.000000       83.500000      11.517000       10      moves=192200

my experiments
   genCea/P1.Pp1000     106.000000      114.099998       5.971227       10      200000          worse           -23.139210
   genCea/P4.Pp1000     111.000000      120.800003       4.984420       10      200000          worse           -29.036762
   genCea/P10.Pp1000    115.000000      121.699997       3.128720       10      200000          worse           -31.565157
   genCea/P1.Pp200       97.000000      102.500000       2.549510       10      200000          worse           -16.019922
   genCea/P1.Pp50        97.000000      105.699997       4.571652       10      200000          worse           -17.502218
   genCea/P1.Ppbest      98.000000      105.800003       3.881580       10      200000          worse           -17.970682
   oneCea/P1.Pp1000     124.000000      134.800003       3.794733       10       60000          worse           -41.457699
   twoCea/P1.Pp1000     133.000000      135.699997       0.948683       10       21000          worse           -46.753319

   genTam/P1.Pp1000      64.000000       68.800003       4.131182       10      200000          better          11.751284
   genTam/P4.Pp1000      64.000000       65.400002       2.988868       10      200000          better          15.028190
   genTam/P10.Pp1000     64.000000       67.400002       4.623611       10      200000          better          12.672604
   genTam/P1.Pp200       64.000000       69.000000       5.906682       10      200000          better          10.984949
   genTam/P1.Pp50        64.000000       70.400002       6.310485       10      200000          better          9.811292
   genTam/P1.Ppbest      64.000000       74.400002       6.022181       10      200000          better          6.871264
   oneTam/P1.Pp1000      64.000000       64.800003       2.529822       10       32000          better          15.778020
   twoTam/P1.Pp1000      64.000000       64.000000       0.000000       10       16000          better          18.170418

   genDK/P1.Pp1000       64.000000       76.199997      10.643725       10      200000          better          4.903781
   genDK/P1.Ppbest       64.000000       78.800003      12.795833       10      200000          better          3.014254
   twoDK/P1.Pp1000       64.000000       64.000000       0.000000       10       16000          better          18.170418

   genTam2/P1.Pp1000     64.000000       64.000000       0.000000       10      200000          better          18.170418
   twoTam2/P1.Pp1000     64.000000       64.800003       2.529822       10       13000          better          15.778020

   genDK2/P1.Pp1000      64.000000       66.400002       5.146736       10      200000          better          13.246767
   twoDK2/P1.Pp1000      64.000000       66.000000       6.324555       10       12000          better          13.101521

   genKad/P1.Pp1000      64.000000       67.800003       5.028806       10      200000          better          12.205501
   genKad/P4.Pp1000      64.000000       67.400002       4.427189       10      200000          better          12.750423
   genKad/P10.Pp1000     64.000000       68.400002       4.788876       10      200000          better          11.825100
   genKad/P1.Pp200       64.000000       68.400002       5.399589       10      200000          better          11.609687
   genKad/P1.Pp50        64.000000       65.199997       3.794733       10      200000          better          14.789006
   oneKad/P1.Pp1000      64.000000       64.000000       0.000000       10       30000          better          18.170418
   twoKad/P1.Pp1000      64.000000       65.599998       3.373096       10       11000          better          14.669128
```

Figure B.3: The performance of GAs for Kea91-16

```
problem = Kea91-20
                       best_score      mean_score       std_dev    sample    eval            diagnosis       t-value
original_paper         153.000000      170.750000       13.679000    10      moves=219024

my experiments
   genCea/P1.Pp1000    181.000000      189.899994        8.748968    10      200000          worse           -12.787148
   genCea/P4.Pp1000    191.000000      203.800003        8.829245    10      200000          worse           -22.029299
   genCea/P10.Pp1000   192.000000      221.100006       14.278578    10      200000          worse           -30.112703
   genCea/P1.Pp200     181.000000      196.699997       11.747813    10      200000          worse           -16.273890
   genCea/P1.Pp50      183.000000      195.500000       10.522463    10      200000          worse           -15.909422
   genCea/P1.Ppbest    198.000000      216.399994       16.077589    10      200000          worse           -26.463614
   oneCea/P1.Pp1000    187.000000      203.800003       12.452577    10       84000          worse           -20.445074
   twoCea/P1.Pp1000    186.000000      209.000000       18.973665    10       45000          worse           -21.167624

   genTam/P1.Pp1000    170.000000      171.000000        0.666667    10      200000          -               -0.208727
   genTam/P4.Pp1000    170.000000      171.600006        0.966092    10      200000          -               -0.702386
   genTam/P10.Pp1000   163.000000      170.199997        3.119829    10      200000          -               0.424351
   genTam/P1.Pp200     163.000000      170.399994        3.025815    10      200000          -               0.270804
   genTam/P1.Pp50      163.000000      168.699997        3.772709    10      200000          -               1.551799
   genTam/P1.Ppbest    163.000000      171.300003        5.313505    10      200000          -               -0.399093
   oneTam/P1.Pp1000    164.000000      168.000000        2.828427    10       34000          better          2.140390
   twoTam/P1.Pp1000    162.000000      167.600006        3.405877    10       19000          better          2.409926

   genDK/P1.Pp1000     164.000000      172.399994        4.427189    10      200000          -               -1.226221
   genDK/P1.Ppbest     166.000000      175.300003        3.683296    10      200000          worse           -3.453092
   twoDK/P1.Pp1000     163.000000      165.100006        1.523884    10       20000          better          4.582315

   genTam2/P1.Pp1000   162.000000      168.699997        3.128720    10      200000          -               1.581248
   twoTam2/P1.Pp1000   160.000000      166.500000        4.648775    10       22000          better          3.139309

   genDK2/P1.Pp1000    164.000000      171.800003        4.022161    10      200000          -               -0.789205
   twoDK2/P1.Pp1000    160.000000      165.199997        3.119829    10       19000          better          4.282070

   genKad/P1.Pp1000    162.000000      168.500000        3.274480    10      200000          -               1.728037
   genKad/P4.Pp1000    163.000000      170.500000        2.677063    10      200000          -               0.195479
   genKad/P10.Pp1000   166.000000      171.600006        3.470511    10      200000          -               -0.649077
   genKad/P1.Pp200     162.000000      168.000000        4.000000    10      200000          -               2.068254
   genKad/P1.Pp50      158.000000      168.000000        5.676462    10      200000          -               1.976655
   oneKad/P1.Pp1000    163.000000      166.399994        3.339993    10       40000          better          3.334440
   twoKad/P1.Pp1000    163.000000      166.500000        2.460804    10       18000          better          3.345336
```

Figure B.4: The performance of GAs for Kea91-20

```
problem = TL91-5
                       best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper         246.820000      246.820000        0.000000       1      CPU=0.32sec

my experiments
   genCea/P1.Pp1000    228.159973      228.159973        0.000003      10      200000          better
   genCea/P4.Pp1000    228.159973      228.159973        0.000003      10      200000          better
   genCea/P10.Pp1000   228.159973      228.159973        0.000003      10      200000          better
   genCea/P1.Pp200     228.159973      228.159973        0.000003      10      200000          better
   genCea/P1.Pp50      228.159973      228.159973        0.000003      10      200000          better
   genCea/P1.Ppbest    228.159973      228.159973        0.000003      10      200000          better
   oneCea/P1.Pp1000    228.159973      228.159973        0.000003      10       32000          better
   twoCea/P1.Pp1000    228.159973      228.159973        0.000003      10       17000          better

   genTam/P1.Pp1000    228.159973      228.159973        0.000003      10      200000          better
   genTam/P4.Pp1000    228.159973      228.159973        0.000003      10      200000          better
   genTam/P10.Pp1000   228.159973      228.159973        0.000003      10      200000          better
   genTam/P1.Pp200     228.159973      228.159973        0.000003      10      200000          better
   genTam/P1.Pp50      228.159973      228.159973        0.000003      10      200000          better
   genTam/P1.Ppbest    228.159973      228.159973        0.000003      10       10000          better
   oneTam/P1.Pp1000    228.159973      228.159973        0.000003      10        6000          better
   twoTam/P1.Pp1000    228.159973      228.159973        0.000003      10        3000          better

   genDK/P1.Pp1000     267.970245      267.970245        0.000004      10      200000          -
   genDK/P1.Ppbest     267.970245      267.970245        0.000004      10       66000          -
   twoDK/P1.Pp1000     267.970245      267.970245        0.000004      10        2000          -

   genTam2/P1.Pp1000   228.159973      228.159973        0.000003      10      200000          better
   twoTam2/P1.Pp1000   228.159973      228.159973        0.000003      10        3000          better

   genDK2/P1.Pp1000    228.159973      235.431549       11.166591      10      200000          better
   twoDK2/P1.Pp1000    228.159973      263.989227       12.589115      10        8000          better

   genKad/P1.Pp1000    228.159973      228.159973        0.000003      10      200000          better
   genKad/P4.Pp1000    228.159973      228.159973        0.000003      10      200000          better
   genKad/P10.Pp1000   228.159973      228.159973        0.000003      10      200000          better
   genKad/P1.Pp200     228.159973      228.159973        0.000003      10      200000          better
   genKad/P1.Pp50      228.159973      228.159973        0.000003      10      200000          better
   oneKad/P1.Pp1000    228.159973      228.159973        0.000003      10        6000          better
   twoKad/P1.Pp1000    228.159973      228.159973        0.000003      10        3000          better
```

Figure B.5: The performance of GAs for TL91-5

```
problem = TL91-6
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper          514.000000      514.000000        0.000000       1      CPU=0.57sec

my experiments
   genCea/P1.Pp1000     361.458984      361.458984        0.000000      10      200000          better
   genCea/P4.Pp1000     361.458984      361.458984        0.000000      10      200000          better
   genCea/P10.Pp1000    361.458984      361.458984        0.000000      10      200000          better
   genCea/P1.Pp200      361.458984      361.458984        0.000000      10      200000          better
   genCea/P1.Pp50       361.458984      361.458984        0.000000      10      200000          better
   genCea/P1.Ppbest     361.458984      385.421112       18.181938      10      200000          better
   oneCea/P1.Pp1000     361.458984      365.385437       12.416529      10       44000          better
   twoCea/P1.Pp1000     361.458984      365.385437       12.416529      10       25000          better

   genTam/P1.Pp1000     377.842407      377.842407        0.000009      10      200000          better
   genTam/P4.Pp1000     377.842407      377.842407        0.000009      10      200000          better
   genTam/P10.Pp1000    377.842407      377.842407        0.000009      10      200000          better
   genTam/P1.Pp200      377.842407      377.842407        0.000009      10      200000          better
   genTam/P1.Pp50       377.842407      377.842407        0.000009      10      200000          better
   genTam/P1.Ppbest     377.842407      377.842407        0.000009      10      200000          better
   oneTam/P1.Pp1000     377.842407      377.842407        0.000009      10        8000          better
   twoTam/P1.Pp1000     377.842407      377.842407        0.000009      10        3000          better

   genDK/P1.Pp1000      377.842407      377.842407        0.000009      10      200000          better
   genDK/P1.Ppbest      377.842407      377.842407        0.000009      10      200000          better
   twoDK/P1.Pp1000      377.842407      377.842407        0.000009      10        3000          better

   genTam2/P1.Pp1000    361.458984      361.458984        0.000000      10      200000          better
   twoTam2/P1.Pp1000    361.458984      376.204071        5.180892      10        8000          better

   genDK2/P1.Pp1000     361.458984      363.097351        5.180892      10      200000          better
   twoDK2/P1.Pp1000     361.458984      376.204071        5.180892      10        7000          better

   genKad/P1.Pp1000     361.458984      361.458984        0.000000      10      200000          better
   genKad/P4.Pp1000     361.458984      366.374023        7.913943      10      200000          better
   genKad/P10.Pp1000    361.458984      368.012360        8.460361      10      200000          better
   genKad/P1.Pp200      361.458984      369.650696        8.634820      10      200000          better
   genKad/P1.Pp50       361.458984      371.289062        8.460361      10      200000          better
   oneKad/P1.Pp1000     361.458984      376.204071        5.180892      10       10000          better
   twoKad/P1.Pp1000     361.458984      372.927399        7.913943      10        7000          better
```

Figure B.6: The performance of GAs for TL91-6

```
problem = TL91-7
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper          559.000000      559.000000        0.000000       1      CPU=4.50sec

my experiments
   genCea/P1.Pp1000     595.895996      595.895996        0.000013      10      200000          -
   genCea/P4.Pp1000     595.895996      606.183960       10.885983      10      200000          -
   genCea/P10.Pp1000    595.895996      611.585632        9.007085      10      200000          -
   genCea/P1.Pp200      595.895996      603.860901       10.391803      10      200000          -
   genCea/P1.Pp50       595.895996      613.896484       10.047390      10      200000          -
   genCea/P1.Ppbest     595.895996      604.381287       10.954443      10      200000          -
   oneCea/P1.Pp1000     595.895996      623.600403       11.971008      10       76000          -
   twoCea/P1.Pp1000     595.895996      621.795166       11.160934      10       28000          -

   genTam/P1.Pp1000     777.573486      777.573486        0.000018      10      200000          -
   genTam/P4.Pp1000     777.573486      777.573486        0.000018      10      200000          -
   genTam/P10.Pp1000    777.573486      777.573486        0.000018      10      200000          -
   genTam/P1.Pp200      777.573486      777.573486        0.000018      10      200000          -
   genTam/P1.Pp50       777.573486      777.573486        0.000018      10      200000          -
   genTam/P1.Ppbest     777.573486      777.573486        0.000018      10      200000          -
   oneTam/P1.Pp1000     777.573486      777.573486        0.000018      10       10000          -
   twoTam/P1.Pp1000     777.573486      777.573486        0.000018      10        4000          -

   genDK/P1.Pp1000      690.411255      690.411255        0.000000      10      200000          -
   genDK/P1.Ppbest      690.411255      747.533020       95.240135      10       22000          -
   twoDK/P1.Pp1000      690.411255      690.411255        0.000000      10        4000          -

   genTam2/P1.Pp1000    595.895996      646.222656       46.741425      10      200000          -
   twoTam2/P1.Pp1000    618.769104      721.910706       61.816608      10       11000          -

   genDK2/P1.Pp1000     595.895996      635.289246       42.626514      10      200000          -
   twoDK2/P1.Pp1000     677.065369      687.272827        4.571360      10        8000          -

   genKad/P1.Pp1000     595.895996      656.832275       34.291699      10      200000          -
   genKad/P4.Pp1000     595.895996      662.478272       36.294914      10      200000          -
   genKad/P10.Pp1000    595.895996      669.981873       39.263294      10      200000          -
   genKad/P1.Pp200      690.411255      690.411255        0.000000      10      200000          -
   genKad/P1.Pp50       618.769104      685.625671       38.084194      10      200000          -
   oneKad/P1.Pp1000     595.895996      679.023865       29.534681      10       18000          -
   twoKad/P1.Pp1000     595.895996      670.173645       39.368294      10        8000          -
```

Figure B.7: The performance of GAs for TL91-7

```
problem = TL91-8
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper          839.000000      839.000000       0.000000       1      CPU=12.45sec

my experiments
   genCea/P1.Pp1000     878.005859      883.760437       6.630581      10      200000            -
   genCea/P4.Pp1000     879.434937      890.575500      12.550374      10      200000            -
   genCea/P10.Pp1000    879.434937      910.045715      18.026203      10      200000            -
   genCea/P1.Pp200      878.005859      891.772278      10.614856      10      200000            -
   genCea/P1.Pp50       883.706055      911.755859      12.853746      10      200000            -
   genCea/P1.Ppbest     879.434937      893.549560      11.715662      10      200000            -
   oneCea/P1.Pp1000     879.434937      900.351929      23.536735      10      112000            -
   twoCea/P1.Pp1000     879.434937      892.936951      12.115487      10       41000            -

   genTam/P1.Pp1000     963.957092      963.957092       0.000000      10      200000            -
   genTam/P4.Pp1000     963.957092      968.524414      14.443170      10      200000            -
   genTam/P10.Pp1000    963.957092      963.957092       0.000000      10      200000            -
   genTam/P1.Pp200      963.957092      973.091736      19.257559      10      200000            -
   genTam/P1.Pp50       963.957092      995.928406      22.062306      10      200000            -
   genTam/P1.Ppbest     963.957092      991.361023      23.585598      10      200000            -
   oneTam/P1.Pp1000     963.957092      963.957092       0.000000      10       14000            -
   twoTam/P1.Pp1000     963.957092      963.957092       0.000000      10        7000            -

   genDK/P1.Pp1000      1186.526489     1186.526489      0.000000      10      200000            -
   genDK/P1.Ppbest      1186.526489     1188.216187      1.454272      10      200000            -
   twoDK/P1.Pp1000      1186.526489     1186.526489      0.000000      10        7000            -

   genTam2/P1.Pp1000    920.892761      936.021912       9.717690      10      200000            -
   twoTam2/P1.Pp1000    963.957092      963.957092       0.000000      10        6000            -

   genDK2/P1.Pp1000     883.706055      954.587341      54.889049      10      200000            -
   twoDK2/P1.Pp1000     883.706055      970.484375     100.118156      10       17000            -

   genKad/P1.Pp1000     920.892761      934.568420       8.995599      10      200000            -
   genKad/P4.Pp1000     942.889343      949.243591       6.698001      10      200000            -
   genKad/P10.Pp1000    920.892761      955.494507      18.651298      10      200000            -
   genKad/P1.Pp200      912.263062      937.001831      15.023054      10      200000            -
   genKad/P1.Pp50       940.171387      947.780518       9.104264      10      200000            -
   oneKad/P1.Pp1000     912.263062      958.787659      16.347078      10       26000            -
   twoKad/P1.Pp1000     963.957092      963.957092       0.000000      10        6000            -
```

Figure B.8: The performance of GAs for TL91-8

```
problem = TL91-12
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper          3162.000000     3162.000000       0.000000       1      CPU=89.50sec

my experiments
   genCea/P1.Pp1000     3332.103272     3614.520020     122.116615      10      200000          -
   genCea/P4.Pp1000     3283.125977     3750.754150     196.572586      10      200000          -
   genCea/P10.Pp1000    3773.731201     4094.222412     177.127136      10      200000          -
   genCea/P1.Pp200      3460.802490     3643.203613      97.281075      10      200000          -
   genCea/P1.Pp50       3617.242920     3860.454346     116.487221      10      200000          -
   genCea/P1.Ppbest     3467.347168     3719.547363     113.399506      10      200000          -
   oneCea/P1.Pp1000     3737.242188     4198.627930     317.858337      10      102000          -
   twoCea/P1.Pp1000     3736.922119     4267.774414     331.191986      10       48000          -

   genTam/P1.Pp1000     3873.534912     3894.586914      66.572327      10      200000          -
   genTam/P4.Pp1000     3873.534912     3873.534912       0.000058      10      200000          -
   genTam/P10.Pp1000    3873.534912     3896.363770      72.191582      10      200000          -
   genTam/P1.Pp200      3873.534912     3951.697998     109.954536      10      200000          -
   genTam/P1.Pp50       3873.534912     4061.785889     143.405243      10      200000          -
   genTam/P1.Ppbest     3873.534912     4015.611816     122.319786      10      200000          -
   oneTam/P1.Pp1000     3873.534912     3873.534912       0.000058      10       28000          -
   twoTam/P1.Pp1000     3873.534912     3873.534912       0.000058      10       12000          -

   genDK/P1.Pp1000      4043.694580     4043.694580       0.000000      10      200000          -
   genDK/P1.Ppbest      4043.694580     4043.694580       0.000000      10      200000          -
   twoDK/P1.Pp1000      4043.694580     4043.694580       0.000000      10       12000          -

   genTam2/P1.Pp1000    3652.710449     3749.451172      52.936153      10      200000          -
   twoTam2/P1.Pp1000    3743.458984     3834.512207      62.832668      10       11000          -

   genDK2/P1.Pp1000     3734.104248     3784.311768      47.818855      10      200000          -
   twoDK2/P1.Pp1000     3501.493408     3779.607666     235.153183      10       26000          -

   genKad/P1.Pp1000     3623.850342     3789.745850      95.616493      10      200000          -
   genKad/P4.Pp1000     3648.143311     3788.478760      79.157371      10      200000          -
   genKad/P10.Pp1000    3483.767090     3785.828369     190.024445      10      200000          -
   genKad/P1.Pp200      3489.833740     3773.408936     127.258614      10      200000          -
   genKad/P1.Pp50       3681.120117     3844.247314     137.852051      10      200000          -
   oneKad/P1.Pp1000     3511.177490     3763.171387     129.534500      10       40000          -
   twoKad/P1.Pp1000     3664.515381     3783.612793      73.797005      10       15000          -
```

Figure B.9: The performance of GAs for TL91-12

```
problem = TL91-15
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper          5862.000000     5862.000000       0.000000       1      CPU=379.63sec

my experiments
   genCea/P1.Pp1000     8317.396484     8924.414062     278.947296      10      200000          -
   genCea/P4.Pp1000     9455.988281     9987.691406     393.367645      10      200000          -
   genCea/P10.Pp1000    9370.442383     10230.452148    561.180542      10      200000          -
   genCea/P1.Pp200      8827.780273     9279.541016     317.584381      10      200000          -
   genCea/P1.Pp50       8945.306641     9516.456055     423.593079      10      200000          -
   genCea/P1.Ppbest     8412.978516     9173.252930     1014.656738     10      200000          -
   oneCea/P1.Pp1000     9984.753906     10966.375977    740.994568      10      110000          -
   twoCea/P1.Pp1000     9089.847656     10609.231445    926.443420      10       85000          -

   genTam/P1.Pp1000     8271.699219     8400.312500     268.935669      10      200000          -
   genTam/P4.Pp1000     8271.699219     8462.156250     171.101303      10      200000          -
   genTam/P10.Pp1000    8271.699219     8320.924805      87.158043      10      200000          -
   genTam/P1.Pp200      8271.699219     8756.325195     342.857513      10      200000          -
   genTam/P1.Pp50       8271.699219     8644.351562     410.608704      10      200000          -
   genTam/P1.Ppbest     8271.699219     8712.568359     407.107117      10      200000          -
   oneTam/P1.Pp1000     8271.699219     8282.394531      17.220694      10       32000          -
   twoTam/P1.Pp1000     8271.699219     8313.614258      84.322571      10       22000          -

   genDK/P1.Pp1000      8940.226562     8946.646484      20.301872      10      200000          -
   genDK/P1.Ppbest      8855.281250     8954.502930     114.988533      10      200000          -
   twoDK/P1.Pp1000      8855.281250     8910.395508      48.101650      10       19000          -

   genTam2/P1.Pp1000    7919.864258     8279.952148     169.844299      10      200000          -
   twoTam2/P1.Pp1000    8212.912109     8276.515625      27.961998      10       20000          -

   genDK2/P1.Pp1000     8545.898438     8632.998047      81.171936      10      200000          -
   twoDK2/P1.Pp1000     8094.010742     8568.381836     278.045898      10       23000          -

   genKad/P1.Pp1000     8042.715820     8629.888672     255.327652      10      200000          -
   genKad/P4.Pp1000     7887.435059     8503.980469     293.119446      10      200000          -
   genKad/P10.Pp1000    7803.100098     8343.980469     314.858124      10      200000          -
   genKad/P1.Pp200      8107.662109     8426.512695     218.564835      10      200000          -
   genKad/P1.Pp50       7383.993164     8180.133789     418.456207      10      200000          -
   oneKad/P1.Pp1000     7864.202637     8245.561523     136.365951      10       42000          -
   twoKad/P1.Pp1000     7383.993164     8267.092773     378.901642      10       24000          -
```

Figure B.10: The performance of GAs for TL91-15

```
problem = TL91-20
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
my experiments
   genCea/P1.Pp1000     18807.140625    21266.957031    1171.740845     10      200000
   genCea/P4.Pp1000     21176.699219    24357.820312    2234.009277     10      200000
   genCea/P10.Pp1000    23422.710938    27848.710938    2215.940918     10      200000
   genCea/P1.Pp200      17358.765625    19100.396484    1073.758545     10      200000
   genCea/P1.Pp50       18408.648438    19466.585938    848.692383      10      200000
   genCea/P1.Ppbest     17306.519531    20112.414062    2084.155273     10      200000
   oneCea/P1.Pp1000     21468.757812    26533.392578    2898.950195     10      142000
   twoCea/P1.Pp1000     22457.214844    28152.285156    3309.413574     10       61000

   genTam/P1.Pp1000     17520.396484    17949.376953    599.628845      10      200000
   genTam/P4.Pp1000     17138.923828    17932.511719    521.343506      10      200000
   genTam/P10.Pp1000    17520.396484    17847.576172    470.604858      10      200000
   genTam/P1.Pp200      17138.923828    18011.208984    669.084778      10      200000
   genTam/P1.Pp50       17138.923828    17710.205078    593.202942      10      200000
   genTam/P1.Ppbest     17138.923828    18071.910156    755.378845      10      200000
   oneTam/P1.Pp1000     17138.923828    17579.396484    192.357529      10       34000
   twoTam/P1.Pp1000     17138.923828    17464.250000    186.537979      10       23000

   genDK/P1.Pp1000      18621.130859    18910.921875    228.993652      10      200000
   genDK/P1.Ppbest      18621.130859    18982.785156    448.364288      10      200000
   twoDK/P1.Pp1000      18621.130859    18756.576172    221.684265      10       18000

   genTam2/P1.Pp1000    16983.781250    17651.339844    467.766296      10      200000
   twoTam2/P1.Pp1000    16534.554688    17088.958984    230.296829      10       36000

   genDK2/P1.Pp1000     17860.634766    18383.845703    282.701202      10      200000
   twoDK2/P1.Pp1000     16795.011719    17771.488281    518.328003      10       33000

   genKad/P1.Pp1000     16983.781250    17495.763672    375.627533      10      200000
   genKad/P4.Pp1000     17246.070312    17835.750000    524.397888      10      200000
   genKad/P10.Pp1000    17704.990234    18293.269531    517.476379      10      200000
   genKad/P1.Pp200      16392.527344    17107.138672    393.399109      10      200000
   genKad/P1.Pp50       16494.166016    17284.037109    490.139587      10      200000
   oneKad/P1.Pp1000     16542.863281    17104.423828    266.851776      10       54000
   twoKad/P1.Pp1000     16569.062500    17048.462891    178.874344      10       35000
```

Figure B.11: The performance of GAs for TL91-20

```
problem = TL91-30
                        best_score      mean_score       std_dev     sample    eval            diagnosis       t-value
my experiments
   genCea/P1.Pp1000     85486.148438    101773.546875   11496.067383    10      200000
   genCea/P4.Pp1000     92526.781250    116348.046875   10629.147461    10      200000
   genCea/P10.Pp1000    116233.914062   130610.359375   7200.506348     10      200000
   genCea/P1.Pp200      51678.230469    59881.246094    6755.524902     10      200000
   genCea/P1.Pp50       54466.183594    58339.425781    4233.992188     10      200000
   genCea/P1.Ppbest     55441.156250    59890.875000    3078.061279     10      200000
   oneCea/P1.Pp1000     84664.695312    121340.156250   16744.035156    10      200000
   twoCea/P1.Pp1000     82052.062500    126090.656250   23802.708984    10      100000

   genTam/P1.Pp1000     53073.984375    55368.816406    1858.915894     10      200000
   genTam/P4.Pp1000     54157.132812    55923.730469    1597.123047     10      200000
   genTam/P10.Pp1000    53981.843750    56721.125000    2239.969482     10      200000
   genTam/P1.Pp200      52801.984375    54348.695312    1321.548340     10      200000
   genTam/P1.Pp50       53949.054688    55878.968750    1975.841675     10      200000
   genTam/P1.Ppbest     54806.390625    57557.792969    1625.220093     10      200000
   oneTam/P1.Pp1000     52750.859375    53366.054688    749.843384      10      200000
   twoTam/P1.Pp1000     52546.523438    53396.937500    1266.951904     10      100000

   genDK/P1.Pp1000      48162.488281    49161.257812    858.077393      10      200000
   genDK/P1.Ppbest      48199.203125    50983.972656    1795.424927     10      200000
   twoDK/P1.Pp1000      46932.183594    47944.675781    814.950623      10       32000

   genTam2/P1.Pp1000    51717.367188    55610.785156    2388.671143     10      200000
   twoTam2/P1.Pp1000    44828.582031    47898.023438    2028.776001     10       89000

   genDK2/P1.Pp1000     49215.300781    50345.437500    844.350281      10      200000
   twoDK2/P1.Pp1000     44366.894531    46025.800781    1273.025757     10       38000

   genKad/P1.Pp1000     49467.511719    50300.664062    698.844788      10      200000
   genKad/P4.Pp1000     49696.132812    51981.894531    2201.885010     10      200000
   genKad/P10.Pp1000    51798.945312    55603.007812    2365.471436     10      200000
   genKad/P1.Pp200      46256.882812    49061.878906    1120.788086     10      200000
   genKad/P1.Pp50       50129.800781    52654.902344    1982.492432     10      200000
   oneKad/P1.Pp1000     41095.054688    45828.257812    1859.938232     10      100000
   twoKad/P1.Pp1000     42814.386719    45472.437500    1533.247070     10       44000
```

Figure B.12: The performance of GAs for TL91-30

```
problem = Tam92-20a
                        best_score      mean_score       std_dev     sample    eval            diagnosis       t-value
original_paper          23544.000000    23544.000000     0.000000      1       trans=1596

my experiments
   genCea/P1.Pp1000     24213.253906    25069.130859    349.714294      10      200000          -
   genCea/P4.Pp1000     24711.658203    25126.824219    350.585144      10      200000          -
   genCea/P10.Pp1000    25058.859375    25420.380859    279.253387      10      200000          -
   genCea/P1.Pp200      22660.625000    24175.535156    843.365418      10      200000          better
   genCea/P1.Pp50       21965.003906    22502.691406    328.097778      10      200000          better
   genCea/P1.Ppbest     21262.339844    23999.666016    1197.599243     10      200000          better
   oneCea/P1.Pp1000     24366.478516    25079.648438    584.135315      10      132000          -
   twoCea/P1.Pp1000     24578.064453    25124.675781    395.743072      10       52000          -

   genTam/P1.Pp1000     21296.617188    21532.310547    236.550385      10      200000          better
   genTam/P4.Pp1000     21030.283203    21446.847656    320.492828      10      200000          better
   genTam/P10.Pp1000    21296.617188    21656.468750    247.886078      10      200000          better
   genTam/P1.Pp200      21175.531250    21434.707031    254.597977      10      200000          better
   genTam/P1.Pp50       21280.443359    21514.419922    186.627640      10      200000          better
   genTam/P1.Ppbest     21296.617188    21608.167969    197.372391      10       28000          better
   oneTam/P1.Pp1000     21030.283203    21387.117188    196.402847      10       50000          better
   twoTam/P1.Pp1000     21230.208984    21303.113281     97.217705      10       18000          better

   genDK/P1.Pp1000      21314.193359    21451.318359    114.821297      10      200000          better
   genDK/P1.Ppbest      21769.216797    22070.160156    207.149948      10       86430          better
   twoDK/P1.Pp1000      20962.078125    21286.972656    238.482422      10       19000          better

   genTam2/P1.Pp1000    20543.183594    21128.865234    347.121429      10      200000          better
   twoTam2/P1.Pp1000    20774.347656    21011.150391    177.241730      10       30000          better

   genDK2/P1.Pp1000     20677.435547    21100.783203    297.608185      10      200000          better
   twoDK2/P1.Pp1000     20687.832031    21152.664062    276.442688      10       26000          better

   genKad/P1.Pp1000     20237.644531    21083.195312    348.583435      10      200000          better
   genKad/P4.Pp1000     21232.371094    21582.121094    236.352676      10      200000          better
   genKad/P10.Pp1000    21030.685547    21557.203125    280.838257      10      200000          better
   genKad/P1.Pp200      20835.195312    21349.921875    250.928680      10      200000          better
   genKad/P1.Pp50       20618.792969    21219.394531    410.436127      10      200000          better
   oneKad/P1.Pp1000     20295.378906    21056.861328    377.643860      10       76000          better
   twoKad/P1.Pp1000     20246.615234    20894.958984    416.154571      10       41000          better
```

Figure B.13: The performance of GAs for Tam92-20a (max. evaluation no. = 20000)

```
problem = Tam92-20a
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper          23544.000000    23544.000000      0.000000       1      trans=1596

my experiments
   genCea/P1.Pp1000     31855.623047    33661.902344    896.988159      10        2000          -
   genCea/P4.Pp1000     32736.091797    35141.250000    1033.993652     10        2000          -
   genCea/P10.Pp1000    32125.931641    36526.722656    2258.483643     10        2000          -
   genCea/P1.Pp200      29927.001953    31779.349609    1279.734863     10        2000          -
   genCea/P1.Pp50       29478.482422    32777.050781    1625.690552     10        2000          -
   genCea/P1.Ppbest     28505.789062    30621.185547    827.815308      10        2000          -
   oneCea/P1.Pp1000     31577.187500    32849.199219    764.832947      10        2000          -
   twoCea/P1.Pp1000     29225.257812    30603.628906    717.476562      10        2000          -

   genTam/P1.Pp1000     23124.921875    23620.154297    283.789825      10        2000          better
   genTam/P4.Pp1000     23561.730469    24464.437500    402.604645      10        2000          -
   genTam/P10.Pp1000    23531.462891    24641.021484    527.763733      10        2000          better
   genTam/P1.Pp200      22717.605469    23737.552734    551.737060      10        2000          better
   genTam/P1.Pp50       22592.845703    23294.076172    444.499023      10        2000          better
   genTam/P1.Ppbest     21457.164062    22032.115234    297.993805      10        2000          better
   oneTam/P1.Pp1000     23101.990234    23399.445312    219.283066      10        2000          better
   twoTam/P1.Pp1000     22620.990234    22941.343750    207.681000      10        2000          better

   genDK/P1.Pp1000      23163.970703    23685.019531    386.171936      10        2000          better
   genDK/P1.Ppbest      21769.216797    22194.820312    388.910553      10        2010          better
   twoDK/P1.Pp1000      22576.789062    23003.312500    297.619721      10        2000          better

   genTam2/P1.Pp1000    22866.203125    23541.681641    421.316010      10        2000          better
   twoTam2/P1.Pp1000    22530.523438    22822.625000    238.947388      10        2000          better

   genDK2/P1.Pp1000     23231.578125    23672.544922    248.678238      10        2000          better
   twoDK2/P1.Pp1000     22336.638672    22819.675781    292.648712      10        2000          better

   genKad/P1.Pp1000     23453.451172    23923.140625    367.561249      10        2000          better
   genKad/P4.Pp1000     23798.248047    24061.525391    204.471863      10        2000          -
   genKad/P10.Pp1000    23969.767578    24686.113281    519.736450      10        2000          -
   genKad/P1.Pp200      23798.248047    24158.630859    246.434708      10        2000          -
   genKad/P1.Pp50       22938.492188    23872.761719    445.877991      10        2000          better
   oneKad/P1.Pp1000     22935.066406    23413.669922    267.260376      10        2000          better
   twoKad/P1.Pp1000     22385.367188    22888.251953    275.551941      10        2000          better
```

Figure B.14: The performance of GAs for Tam92-20a (max. evaluation no. = 2000)

```
problem = Tam92-30a
                        best_score      mean_score       std_dev     sample    eval            diagnosis       t-value
original_paper          45044.000000    45044.000000     0.000000       1      trans=3712

my experiments
   genCea/P1.Pp1000     51158.300781    52672.199219    818.467651      10      200000          -
   genCea/P4.Pp1000     52820.800781    54657.953125    1182.705444     10      200000          -
   genCea/P10.Pp1000    54448.503906    56555.550781    1241.639893     10      200000          -
   genCea/P1.Pp200      47106.113281    49051.128906    1546.246582     10      200000          -
   genCea/P1.Pp50       47897.507812    50463.722656    3210.273682     10      200000          -
   genCea/P1.Ppbest     48646.285156    52795.445312    2203.371094     10      200000          -
   oneCea/P1.Pp1000     51276.636719    54253.718750    2566.256103     10      140000          -
   twoCea/P1.Pp1000     52008.777344    53851.984375    2111.480713     10      100000          -

   genTam/P1.Pp1000     48718.304688    49162.656250    390.288422      10      200000          -
   genTam/P4.Pp1000     49221.429688    50032.550781    475.434631      10      200000          -
   genTam/P10.Pp1000    49006.441406    49927.207031    674.580139      10      200000          -
   genTam/P1.Pp200      48706.585938    49312.136719    414.930969      10      200000          -
   genTam/P1.Pp50       48400.222656    49467.976562    703.621399      10      200000          -
   genTam/P1.Ppbest     48653.011719    49088.195312    419.486145      10      200000          -
   oneTam/P1.Pp1000     48220.554688    48907.714844    388.841003      10      200000          -
   twoTam/P1.Pp1000     47963.789062    48432.757812    332.427612      10      100000          -

   genDK/P1.Pp1000      45233.625000    45866.410156    453.864044      10      200000          -
   genDK/P1.Ppbest      44953.539062    45894.843750    777.329346      10      200000          better
   twoDK/P1.Pp1000      44868.316406    45370.863281    474.107269      10      100000          better

   genTam2/P1.Pp1000    43654.707031    45884.941406    1334.490845     10      200000          better
   twoTam2/P1.Pp1000    44261.878906    45101.398438    700.840759      10       60000          better

   genDK2/P1.Pp1000     45240.769531    45802.429688    540.653992      10      200000          -
   twoDK2/P1.Pp1000     43265.125000    44629.992188    736.882935      10       46000          better

   genKad/P1.Pp1000     45387.050781    45853.375000    362.973755      10      200000          -
   genKad/P4.Pp1000     44747.617188    45999.949219    571.162659      10      200000          better
   genKad/P10.Pp1000    46347.281250    47034.101562    582.116638      10      200000          -
   genKad/P1.Pp200      44491.707031    45434.558594    532.284668      10      200000          better
   genKad/P1.Pp50       45788.089844    46407.089844    523.966858      10      200000          -
   oneKad/P1.Pp1000     43460.730469    44380.507812    528.305725      10       86000          better
   twoKad/P1.Pp1000     43155.996094    44436.238281    959.974670      10       55000          better
```

Figure B.15: The performance of GAs for Tam92-30a (max. evaluation no. = 20000)

```
problem = Tam92-30a
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper          45044.000000    45044.000000      0.000000      1       trans=3712

my experiments
   genCea/P1.Pp1000     73539.320312    77450.554688    1684.140137     10        4000          -
   genCea/P4.Pp1000     76728.679688    79448.156250    2262.581055     10        4000          -
   genCea/P10.Pp1000    79412.804688    82203.656250    2065.252686     10        4000          -
   genCea/P1.Pp200      67955.617188    72453.609375    2663.597900     10        4000          -
   genCea/P1.Pp50       65377.585938    69759.148438    3002.537109     10        4000          -
   genCea/P1.Ppbest     60923.597656    63899.664062    1915.217285     10        4000          -
   oneCea/P1.Pp1000     70294.156250    71664.593750    971.190613      10        4000          -
   twoCea/P1.Pp1000     63642.039062    65682.804688    1591.772217     10        4000          -

   genTam/P1.Pp1000     53632.164062    55036.347656    700.473633      10        4000          -
   genTam/P4.Pp1000     53564.601562    55437.984375    938.200806      10        4000          -
   genTam/P10.Pp1000    54581.175781    55543.644531    735.885071      10        4000          -
   genTam/P1.Pp200      53564.835938    54507.316406    644.056702      10        4000          -
   genTam/P1.Pp50       50869.488281    52731.050781    1182.785278     10        4000          -
   genTam/P1.Ppbest     49005.667969    50154.984375    800.882446      10        4000          -
   oneTam/P1.Pp1000     51276.179688    52603.542969    643.901916      10        4000          -
   twoTam/P1.Pp1000     49414.230469    51393.363281    968.793213      10        4000          -

   genDK/P1.Pp1000      50759.863281    51936.148438    756.871033      10        4000          -
   genDK/P1.Ppbest      45890.761719    47433.605469    748.406799      10        4000          -
   twoDK/P1.Pp1000      47256.804688    48483.460938    628.097778      10        4000          -

   genTam2/P1.Pp1000    52968.859375    54520.914062    1071.784424     10        4000          -
   twoTam2/P1.Pp1000    50765.515625    51360.847656    323.580963      10        4000          -

   genDK2/P1.Pp1000     50649.066406    51920.917969    704.008057      10        4000          -
   twoDK2/P1.Pp1000     47783.054688    48714.242188    706.843933      10        4000          -

   genKad/P1.Pp1000     49847.644531    51680.027344    1241.504028     10        4000          -
   genKad/P4.Pp1000     51417.667969    53217.421875    887.877258      10        4000          -
   genKad/P10.Pp1000    53326.164062    54535.304688    650.998047      10        4000          -
   genKad/P1.Pp200      48966.816406    51900.515625    1235.278564     10        4000          -
   genKad/P1.Pp50       47408.484375    50369.605469    1604.940552     10        4000          -
   oneKad/P1.Pp1000     49070.566406    50837.773438    907.913635      10        4000          -
   twoKad/P1.Pp1000     48153.984375    49015.257812    626.425232      10        4000          -
```

Figure B.16: The performance of GAs for Tam92-30a (max. evaluation no. = 4000)

```
problem = VCea91-10
                        best_score      mean_score        std_dev     sample    eval            diagnosis       t-value
original_paper          24445.000000    24445.000000      0.000000       1      CPU=847sec

my experiments
   genCea/P1.Pp1000     21964.396484    24440.667969    1673.259277     10      200000          better
   genCea/P4.Pp1000     24404.576172    26145.173828    1162.896362     10      200000          better
   genCea/P10.Pp1000    24937.363281    27898.125000    2129.160400     10      200000          -
   genCea/P1.Pp200      22358.863281    24774.328125    1814.364258     10      200000          better
   genCea/P1.Pp50       23082.482422    26614.316406    1887.264648     10      200000          better
   genCea/P1.Ppbest     20385.244141    22575.462891    1396.756836     10      200000          better
   oneCea/P1.Pp1000     21601.291016    22144.529297    427.987000      10       86000          better
   twoCea/P1.Pp1000     19946.144531    22902.320312    1429.428833     10       53000          better

   genTam/P1.Pp1000     22025.931641    22074.931641     34.207214      10      200000          better
   genTam/P4.Pp1000     22025.931641    22047.650391     34.970451      10      200000          better
   genTam/P10.Pp1000    22025.931641    22025.931641      0.000000      10      200000          better
   genTam/P1.Pp200      22025.931641    22181.244141    168.558777      10      200000          better
   genTam/P1.Pp50       22025.931641    22193.767578    189.191773      10      200000          better
   genTam/P1.Ppbest     22025.931641    22129.966797    165.923263      10      200000          better
   oneTam/P1.Pp1000     22025.931641    22025.931641      0.000000      10       18000          better
   twoTam/P1.Pp1000     22025.931641    22025.931641      0.000000      10       11000          better

   genDK/P1.Pp1000      23034.728516    23107.029297    228.634598      10      200000          better
   genDK/P1.Ppbest      23034.728516    23067.923828    104.971382      10      200000          better
   twoDK/P1.Pp1000      23034.728516    23034.728516      0.000000      10        8000          better

   genTam2/P1.Pp1000    20601.035156    21622.613281    539.079285      10      200000          better
   twoTam2/P1.Pp1000    21716.378906    22002.132812    102.956802      10       11000          better

   genDK2/P1.Pp1000     20671.429688    21506.328125    390.307983      10      200000          better
   twoDK2/P1.Pp1000     21125.765625    22094.814453    728.395264      10       15000          better

   genKad/P1.Pp1000     20546.755859    21538.619141    607.798401      10      200000          better
   genKad/P4.Pp1000     20546.755859    21570.591797    485.349091      10      200000          better
   genKad/P10.Pp1000    20853.019531    21470.673828    447.050690      10      200000          better
   genKad/P1.Pp200      20601.035156    21550.281250    619.727295      10      200000          better
   genKad/P1.Pp50       21643.208984    21843.056641    196.747330      10      200000          better
   oneKad/P1.Pp1000     20601.035156    21883.441406    450.591797      10       42000          better
   twoKad/P1.Pp1000     21412.503906    21933.632812    207.344818      10       11000          better
```

Figure B.17: The performance of GAs for VCea91-10

Appendix C

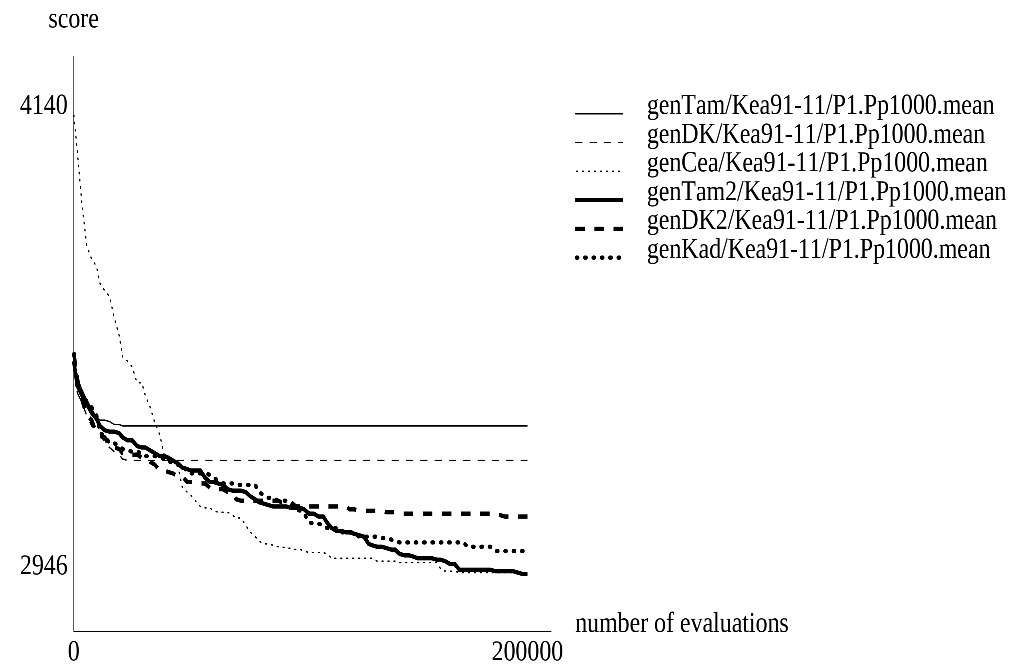
The Results of Algorithm Comparison

This appendix shows the state of convergence of each GA. The name of each GA is denoted as follows.

rrraaa/*fff*/P*ggg*.Pp*ppp*

where *rrr* =	reproduction method (gen, one, two)
aaa =	algorithm (Cea, Tam, DK, Tam2, DK2, Kad)
fff =	the name of FLP (Kea91-11, TL91-5, etc.)
ggg =	the number of populations (1, 4, 10)
ppp =	population size $\times$ the number of populations (50, 200, 1000)

The horizontal axis indicates the number of evaluations, whereas the vertical axis indicates the mean of the best individual scores. Here, smaller score is better.



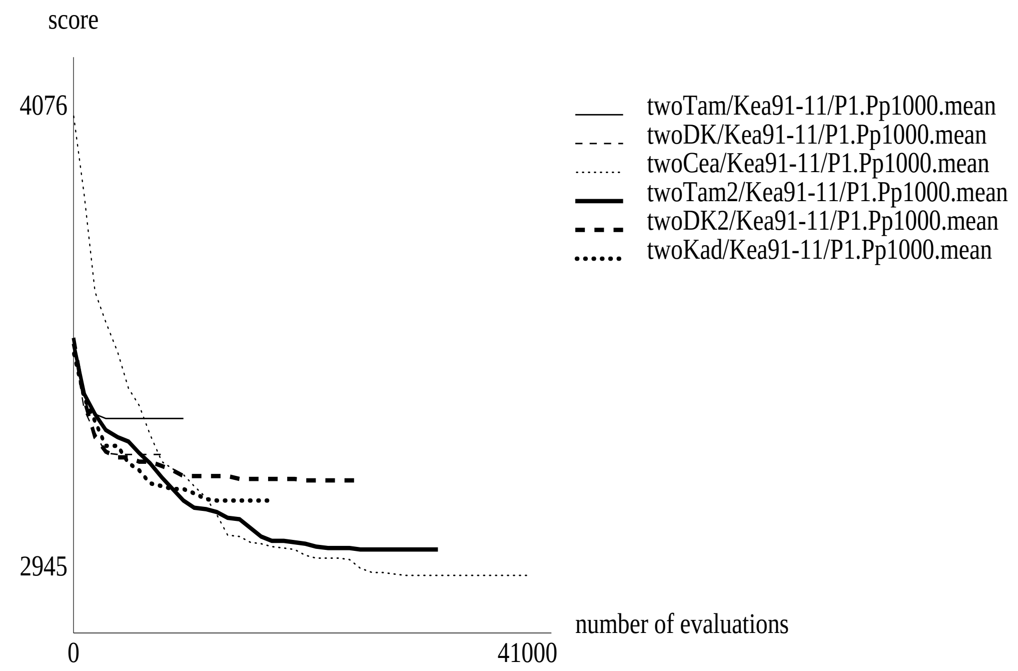
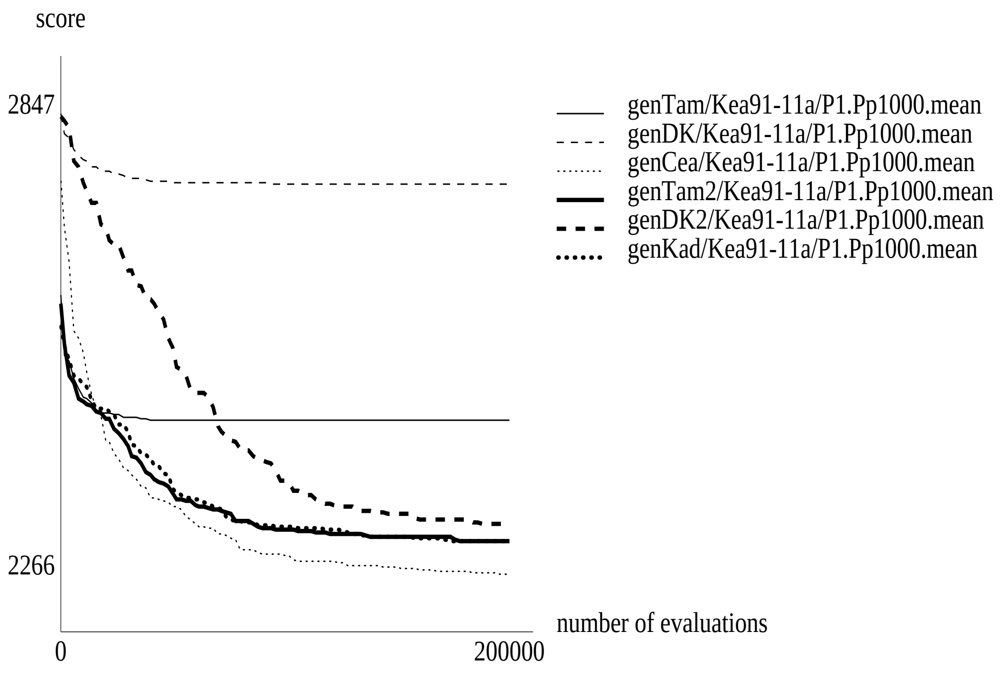


Figure C.1. A comparison of each algorithm in Kea91-11



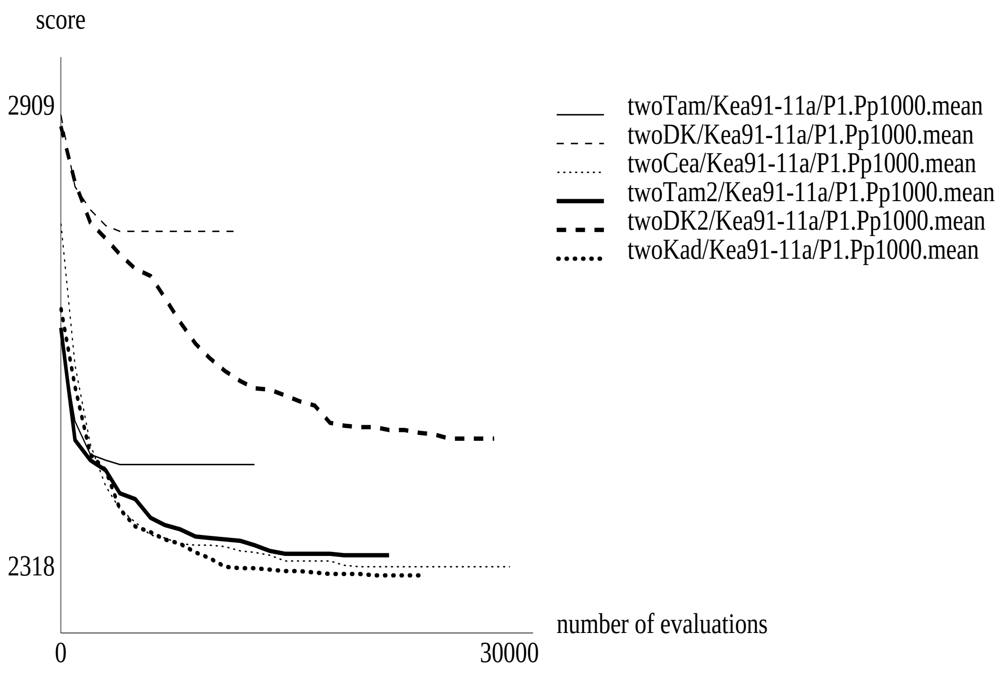
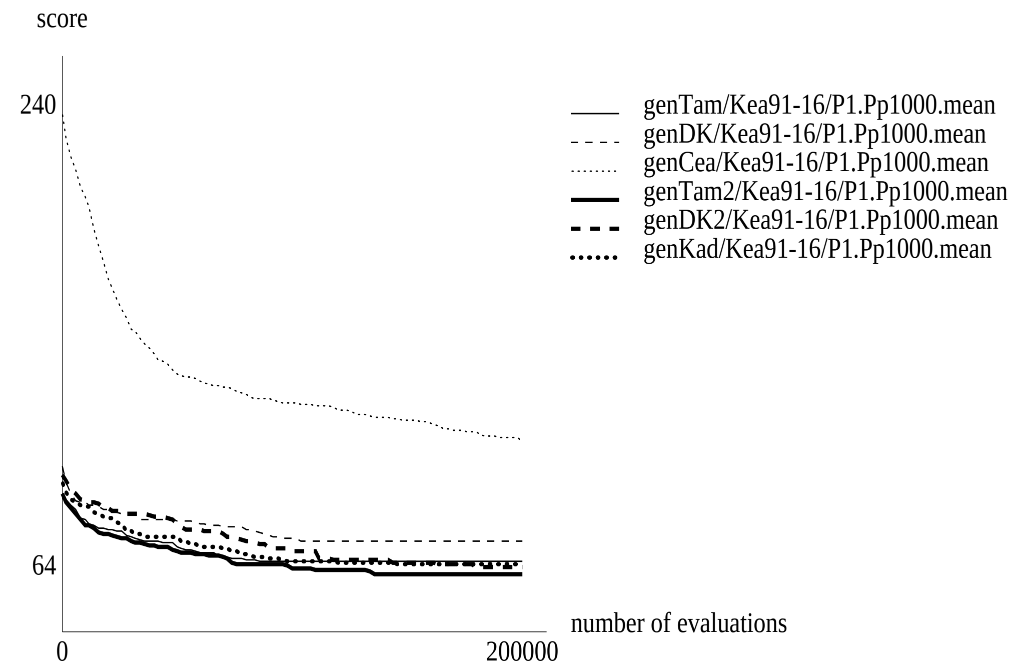


Figure C.2. A comparison of each algorithm in Kea91-11a



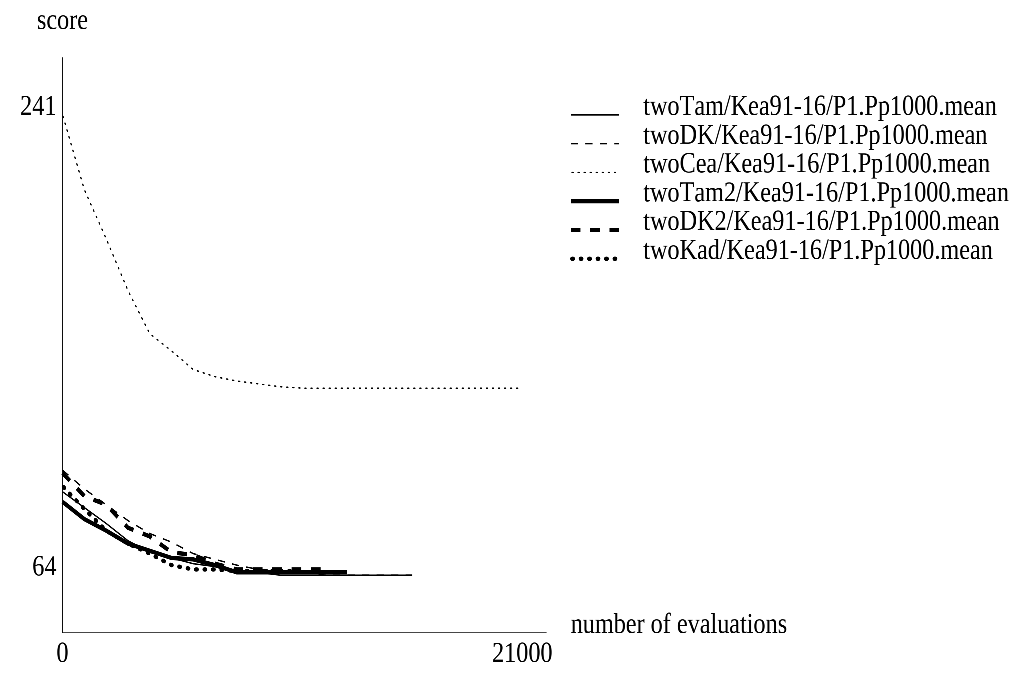
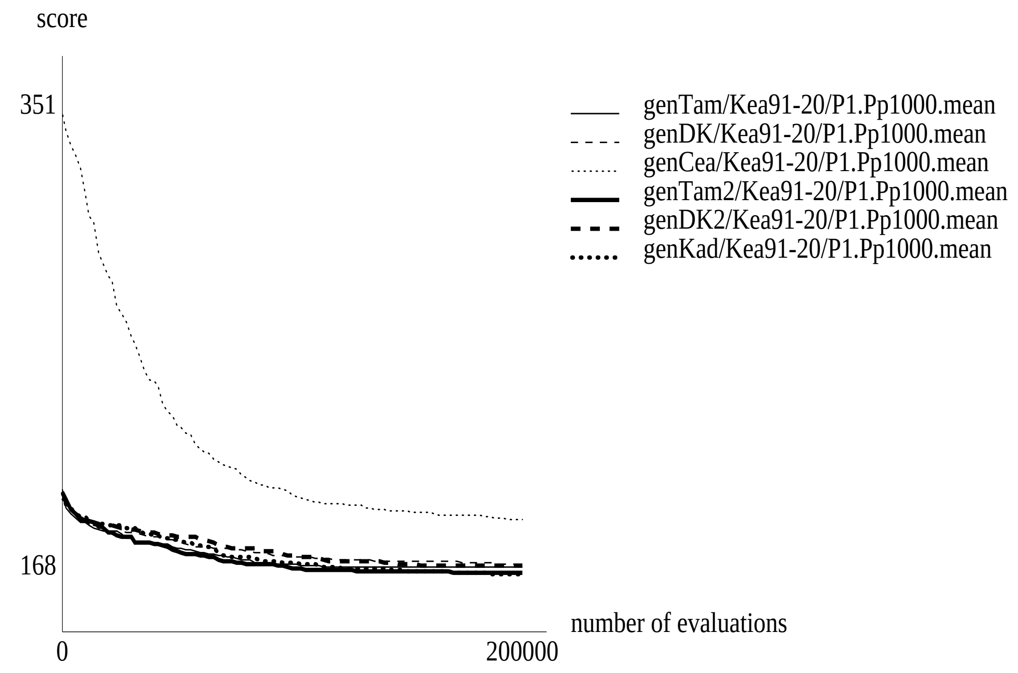


Figure C.3. A comparison of each algorithm in Kea91-16



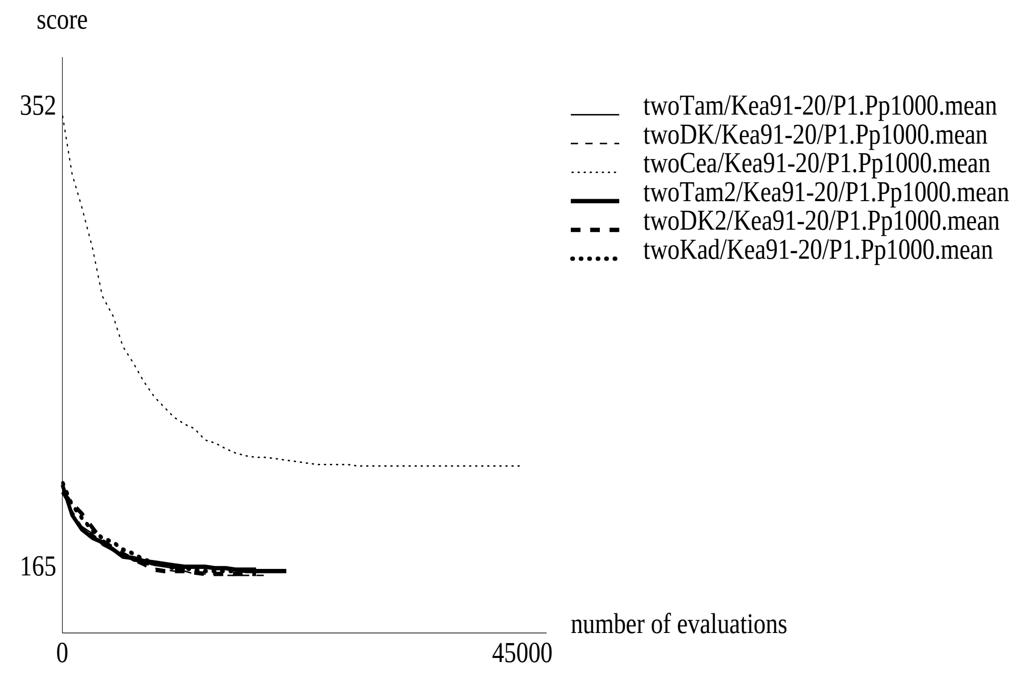
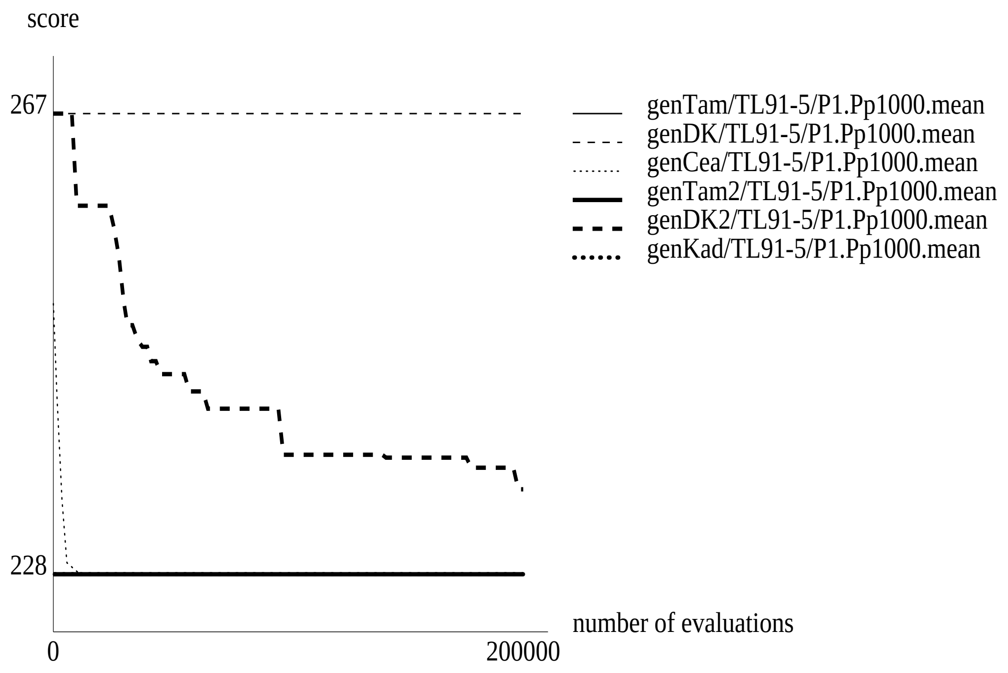


Figure C.4. A comparison of each algorithm in Kea91-20



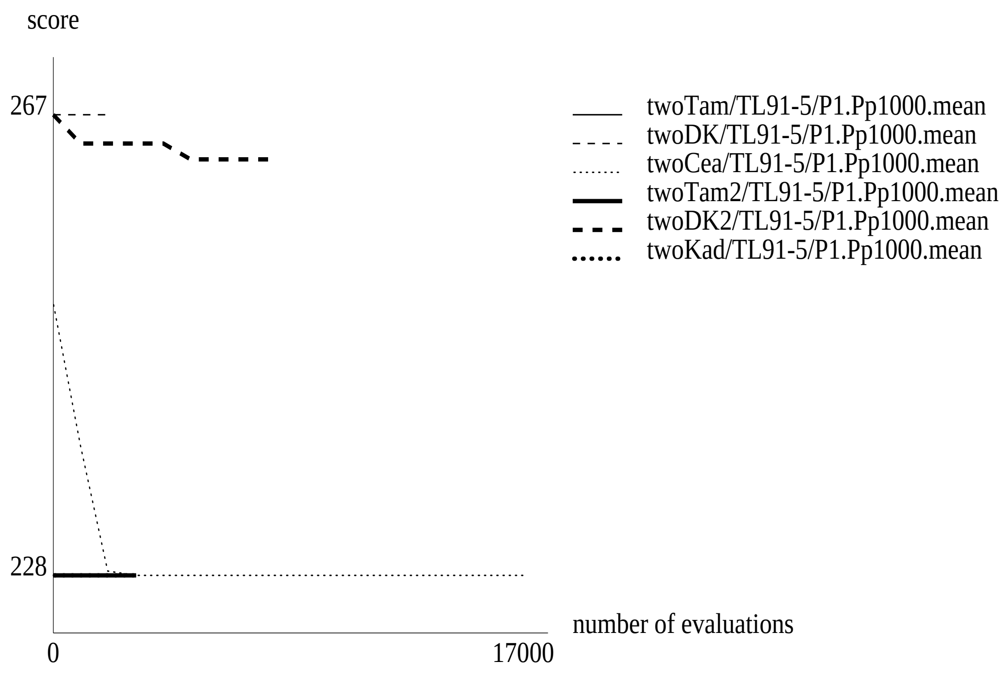
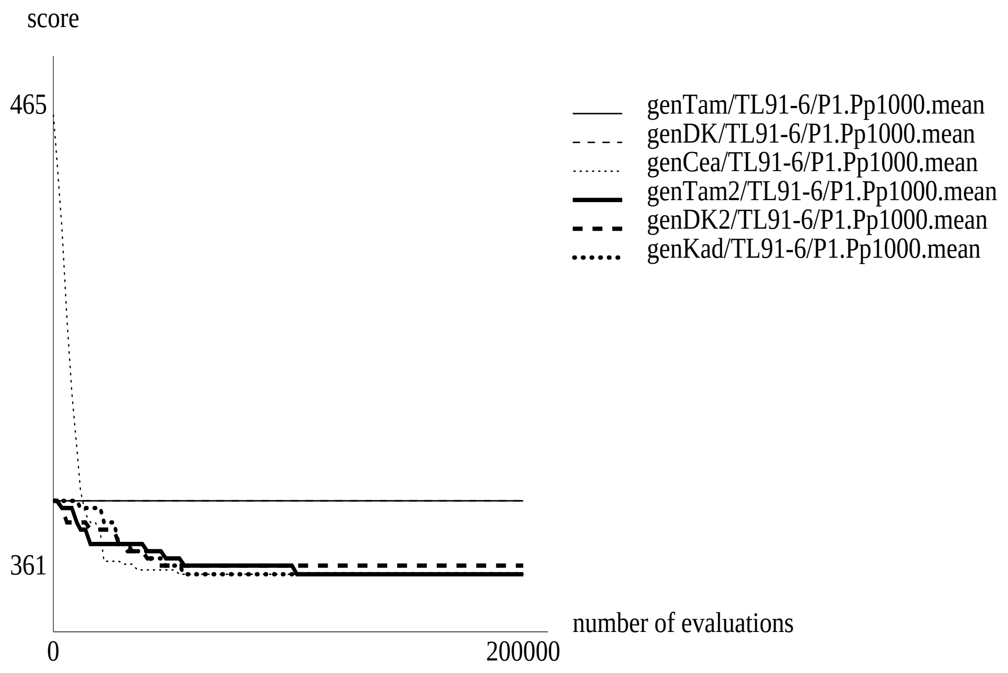


Figure C.5. A comparison of each algorithm in TL91-5



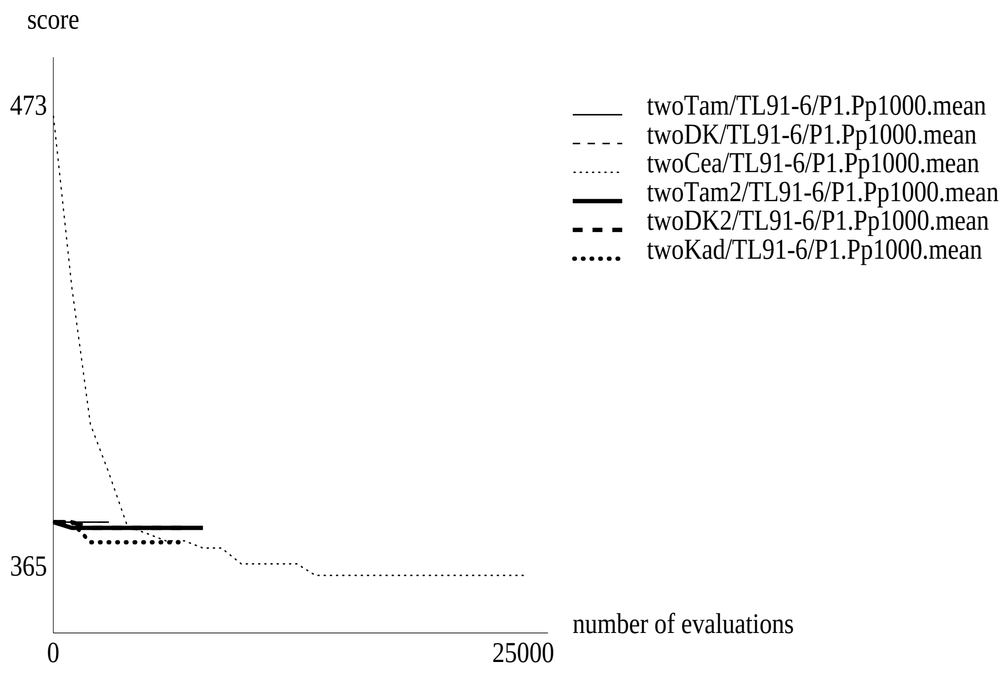
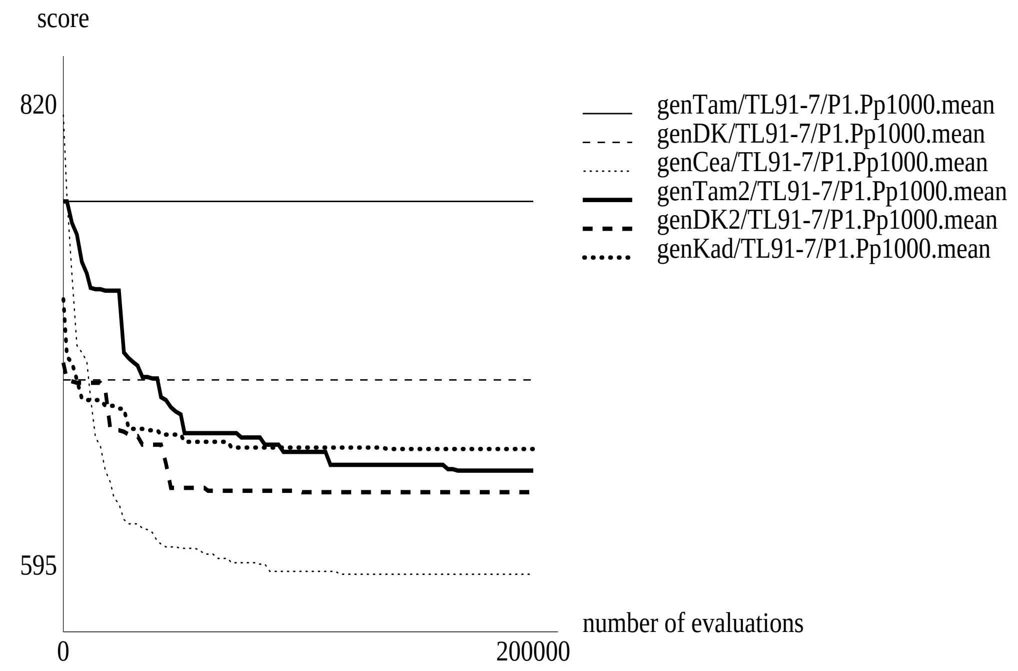


Figure C.6. A comparison of each algorithm in TL91-6



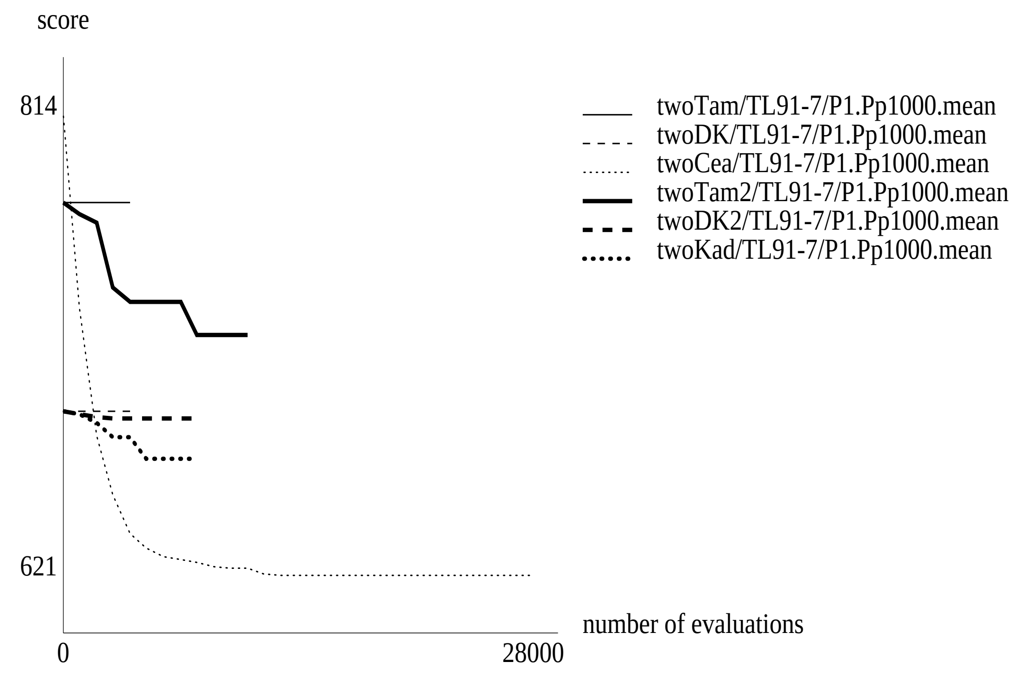
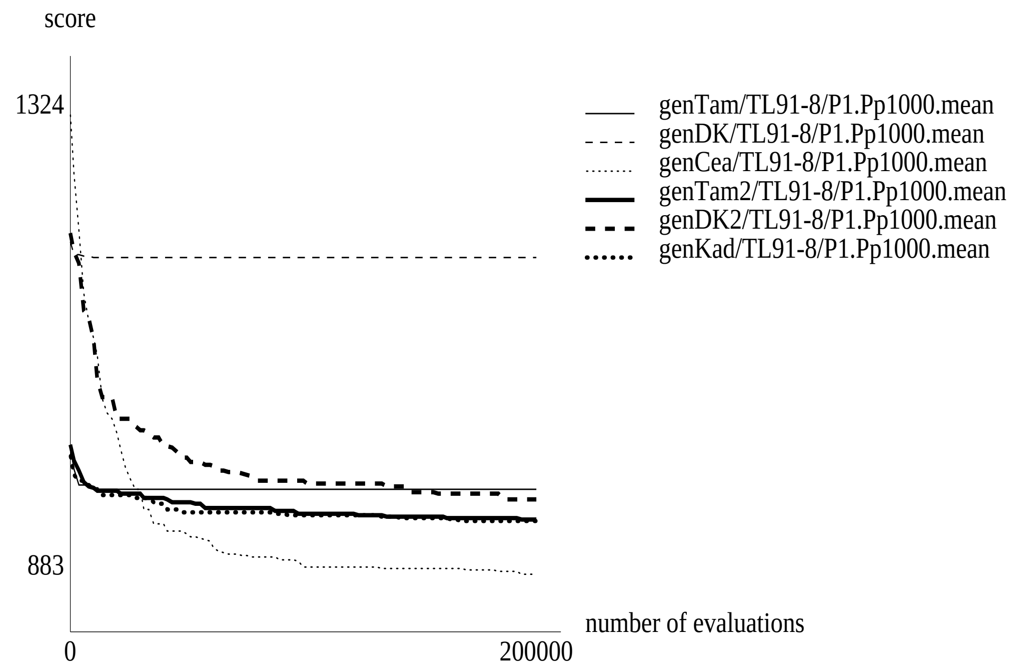


Figure C.7. A comparison of each algorithm in TL91-7



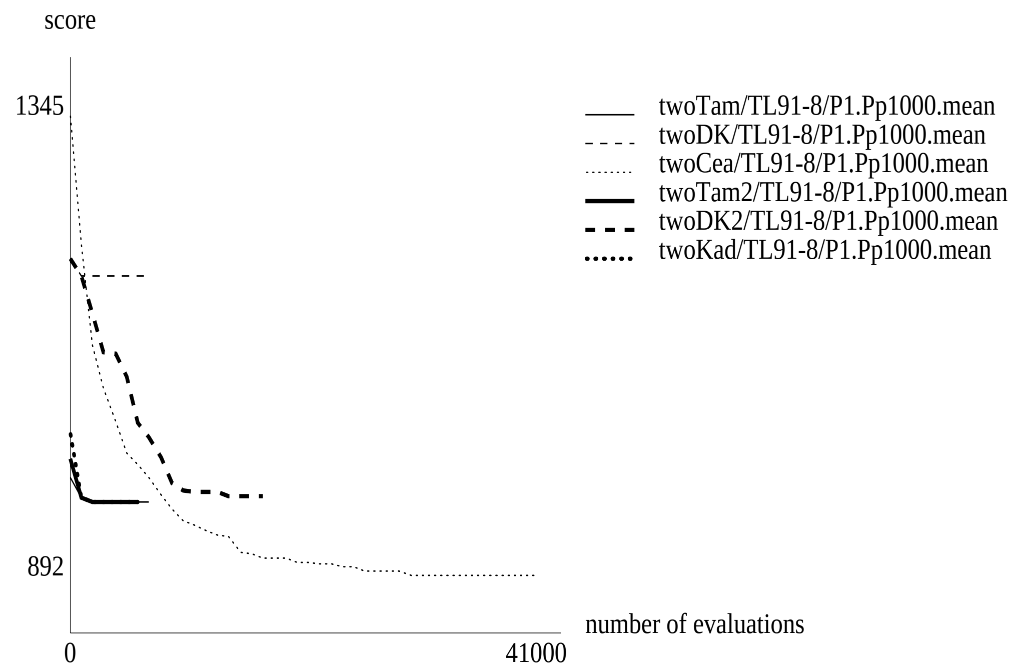
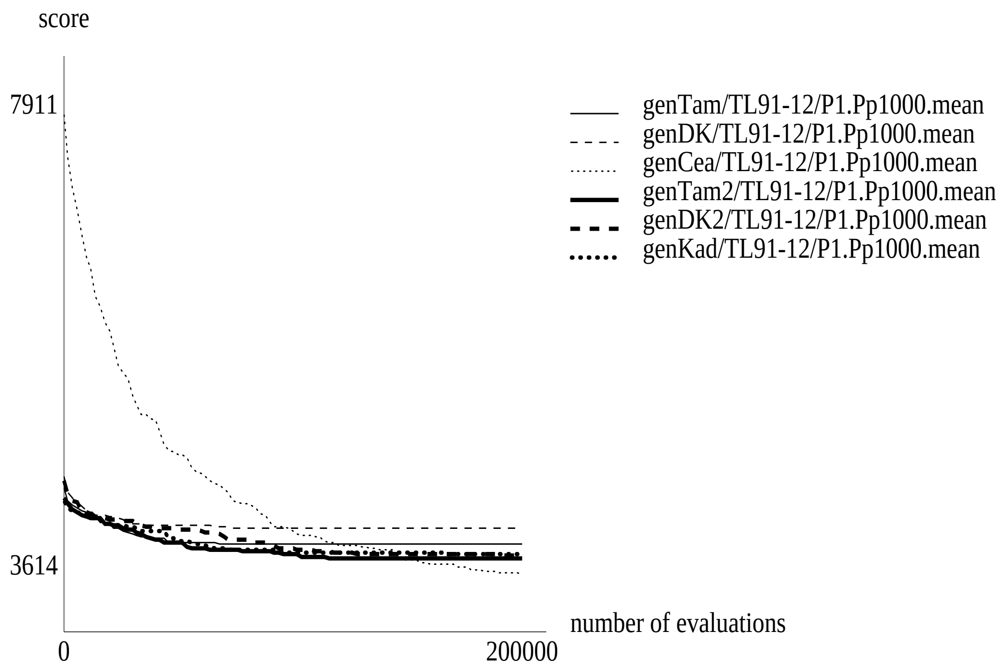


Figure C.8. A comparison of each algorithm in TL91-8



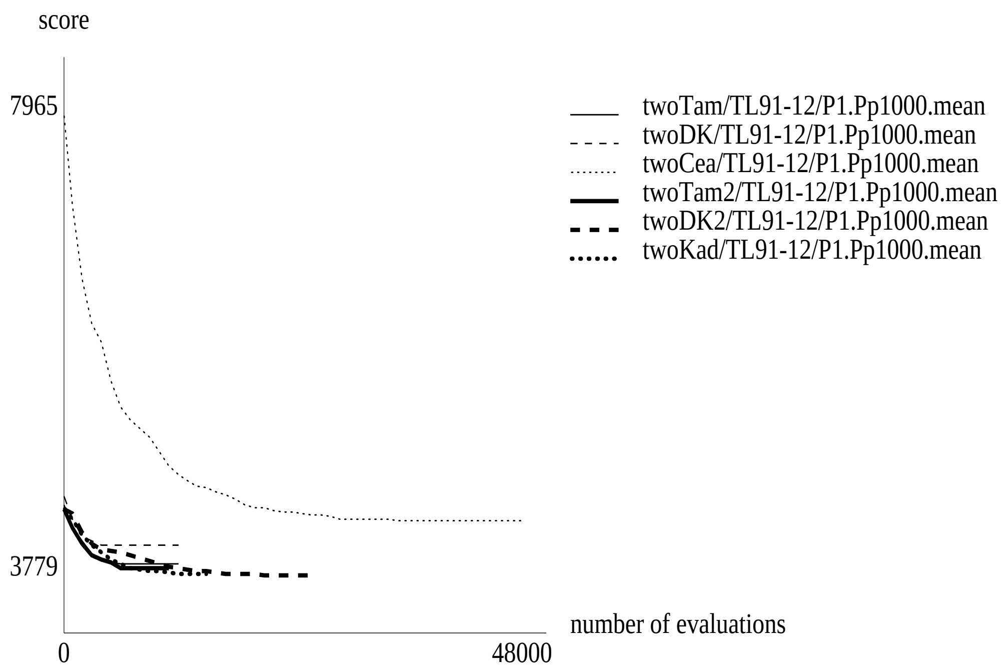
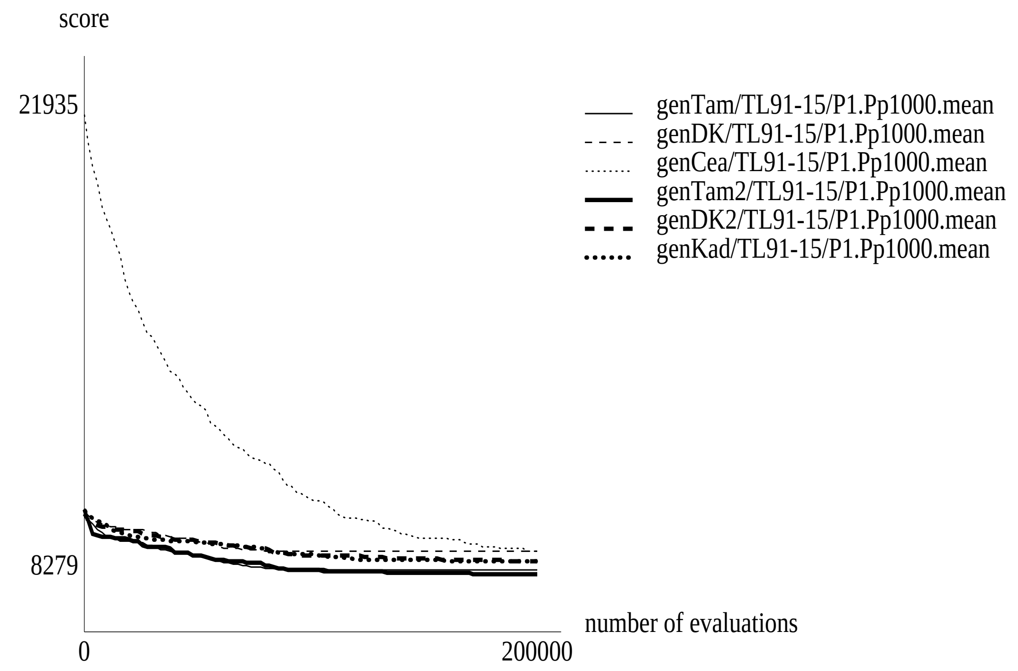


Figure C.9. A comparison of each algorithm in TL91-12



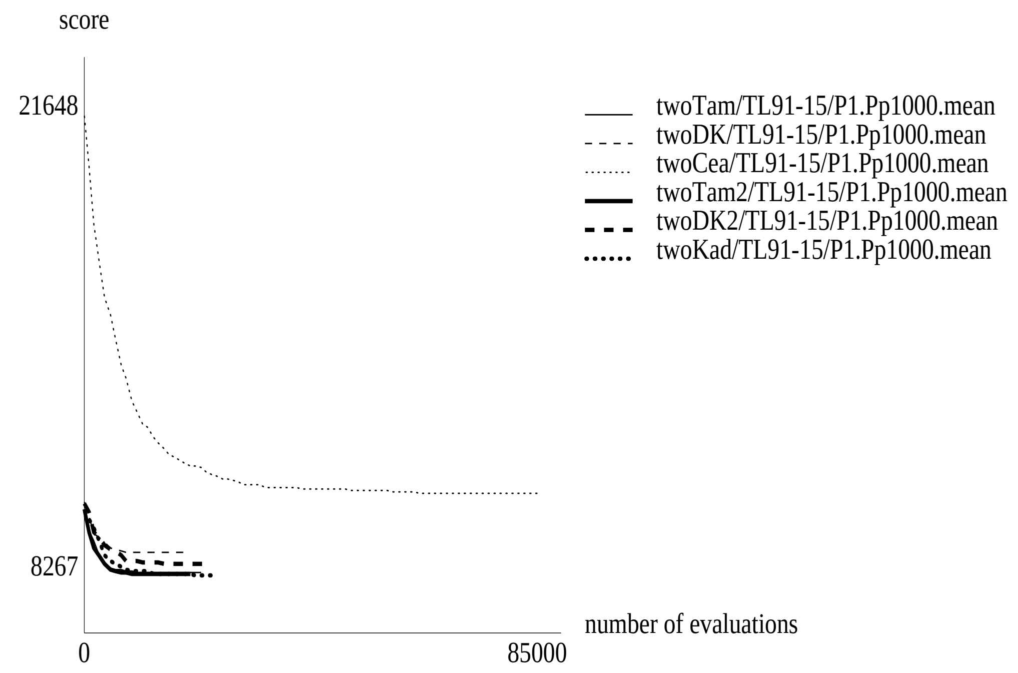
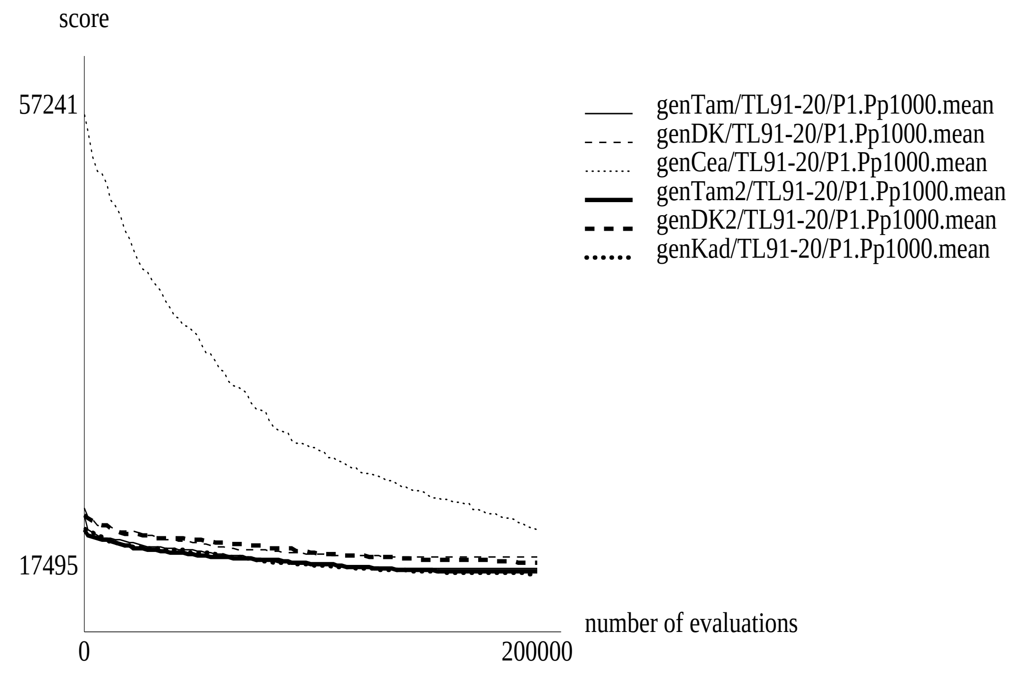


Figure C.10. A comparison of each algorithm in TL91-15



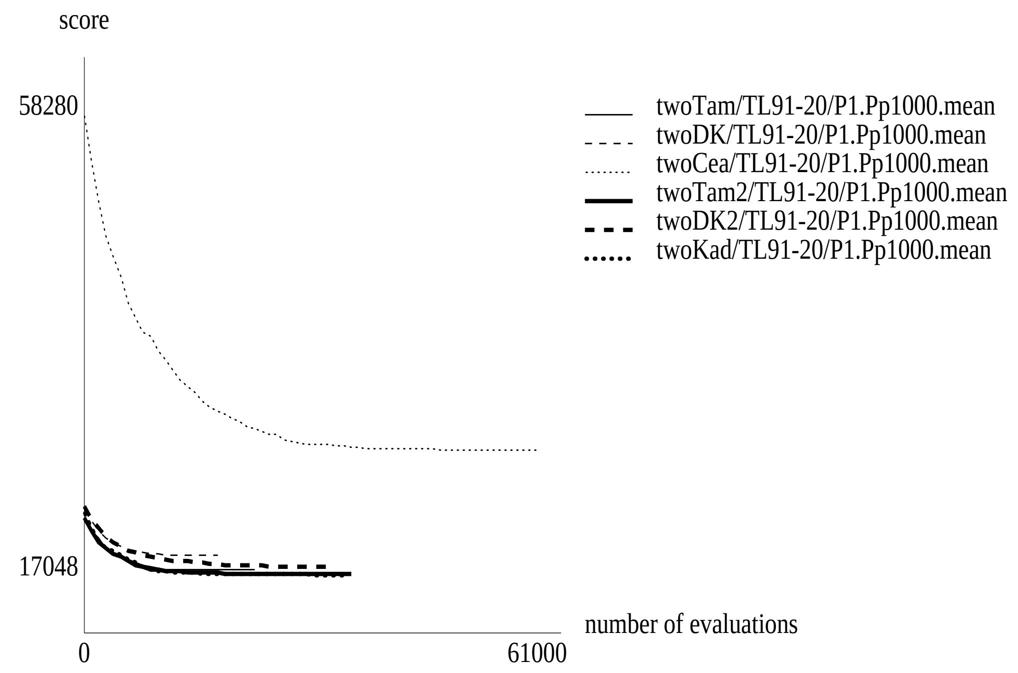
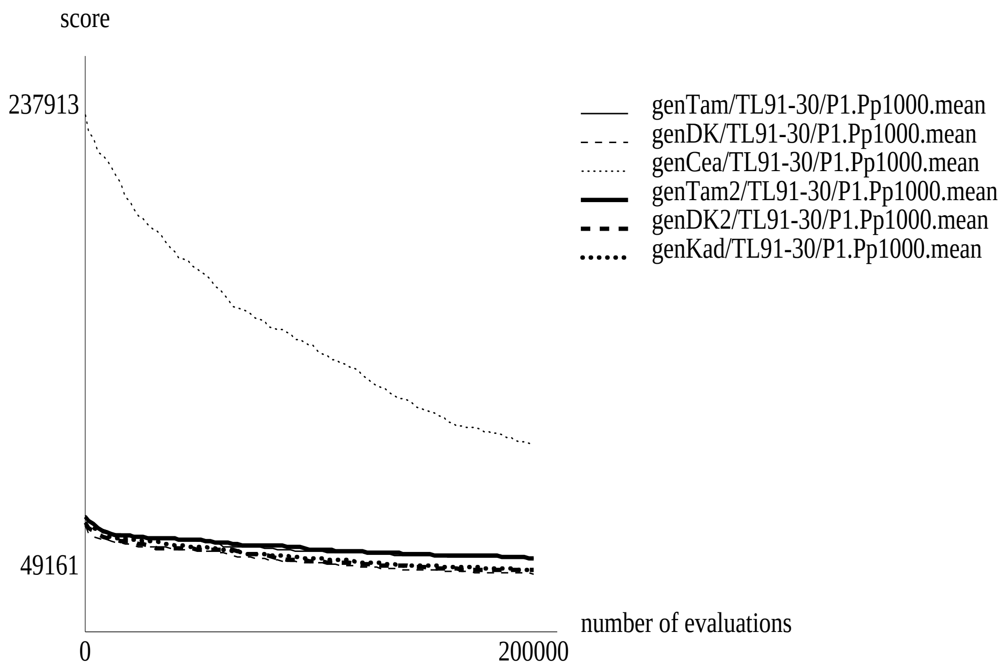


Figure C.11. A comparison of each algorithm in TL91-20



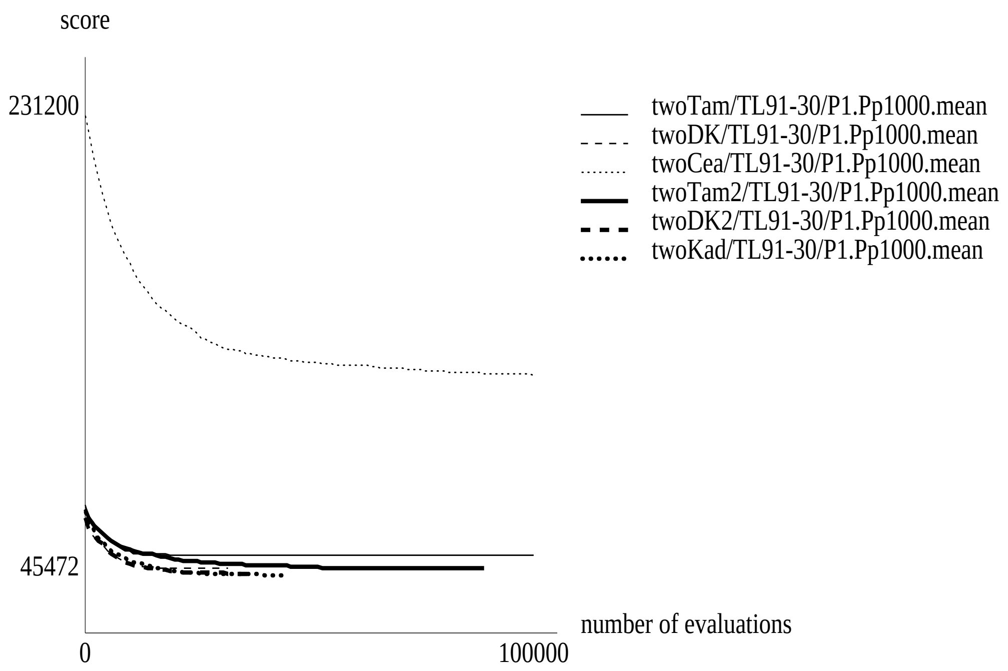
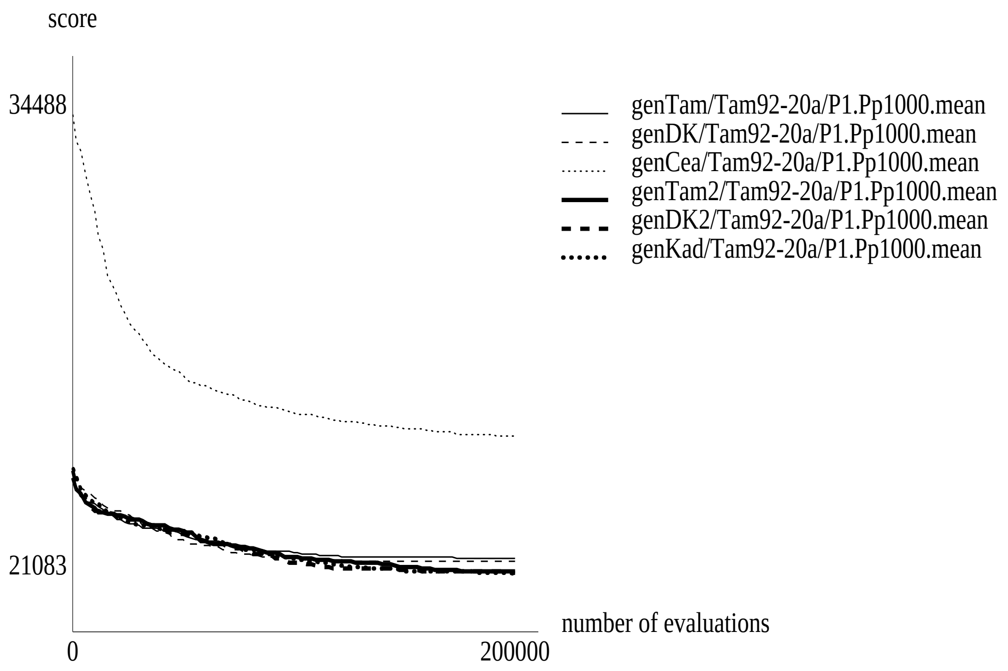


Figure C.12. A comparison of each algorithm in TL91-30



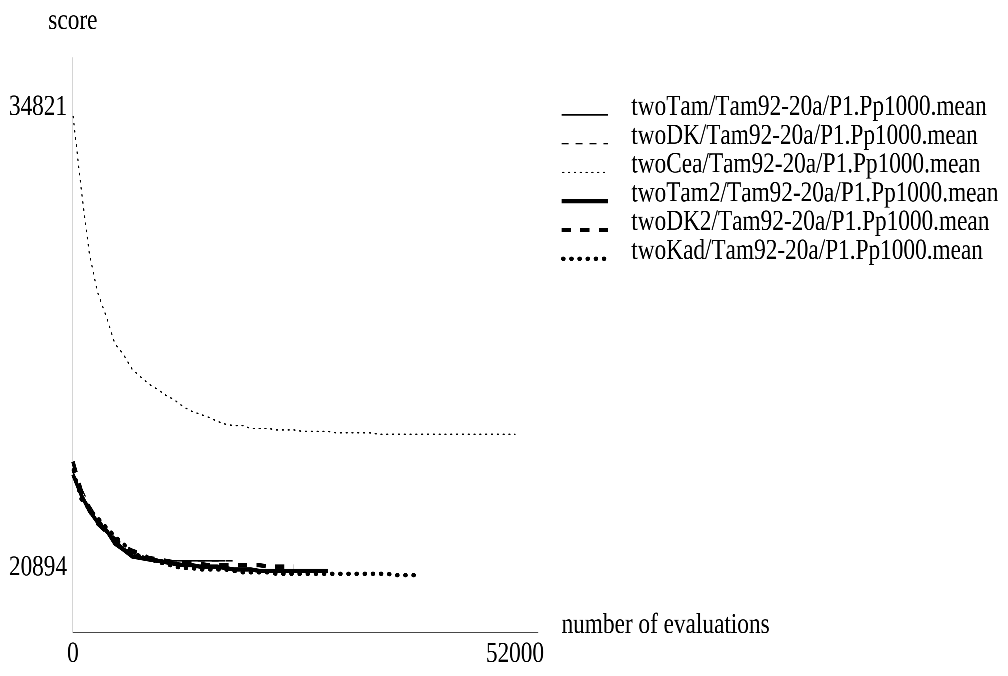
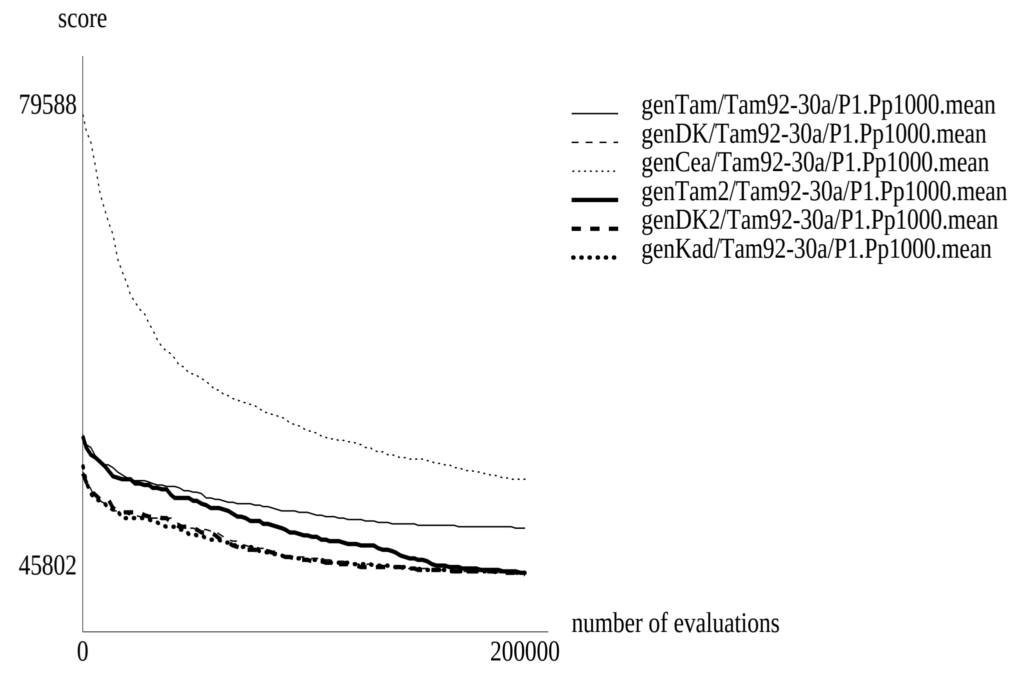


Figure C.13. A comparison of each algorithm in Tam92-20a



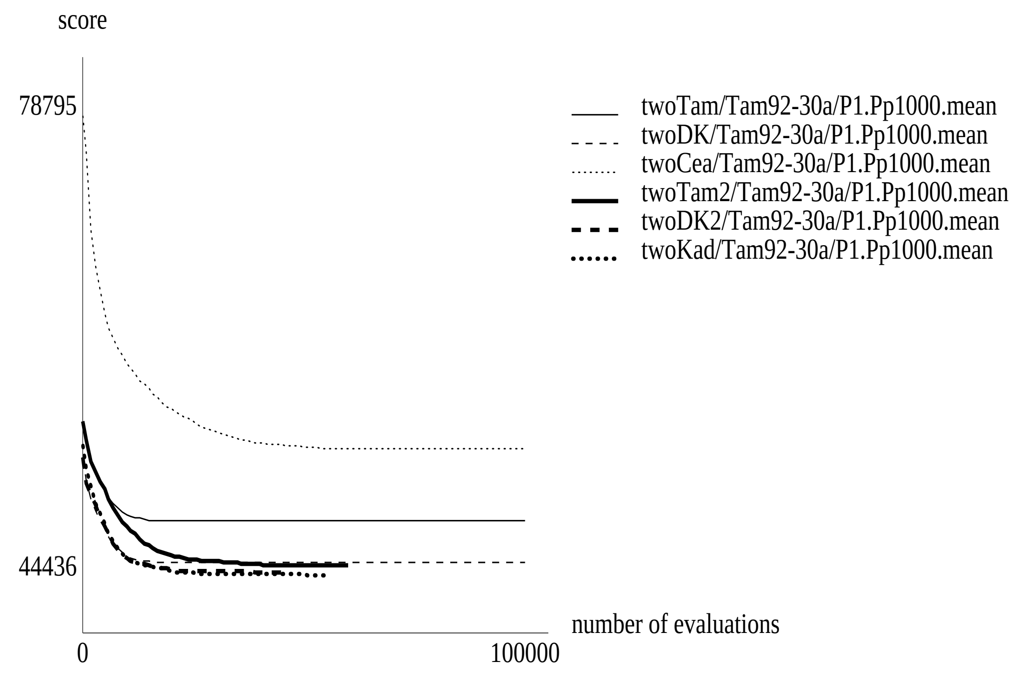
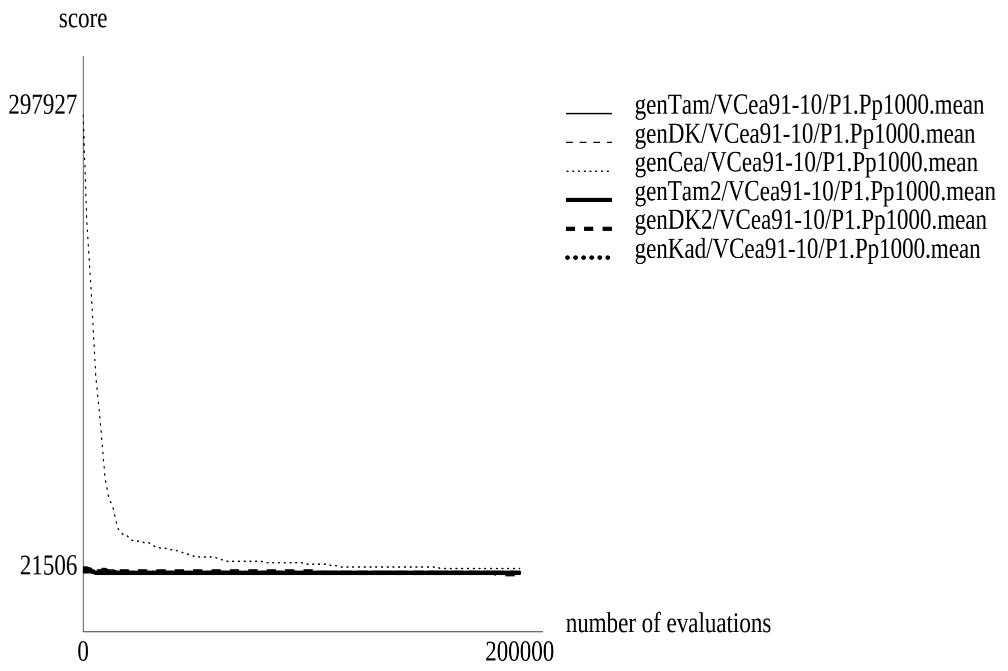


Figure C.14. A comparison of each algorithm in Tam92-30a



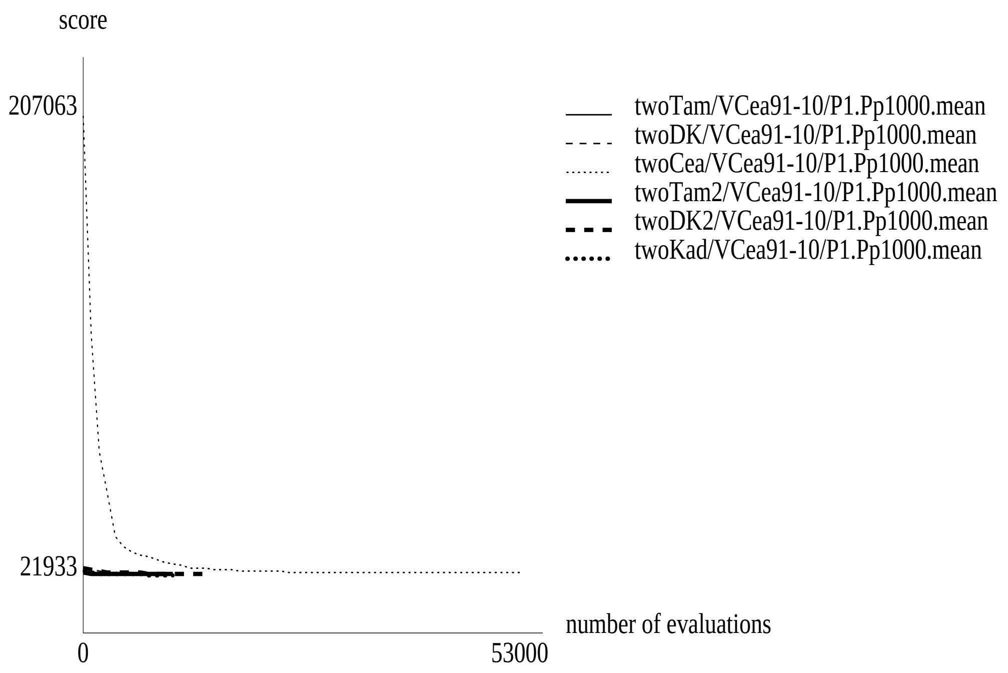


Figure C.15. A comparison of each algorithm in VCea91-10

Appendix D

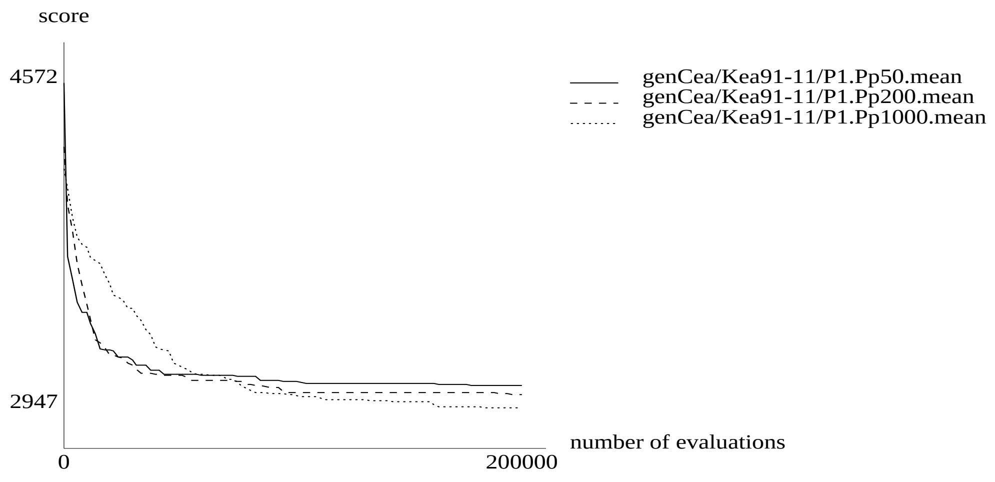
The Results of Population Size Investigation

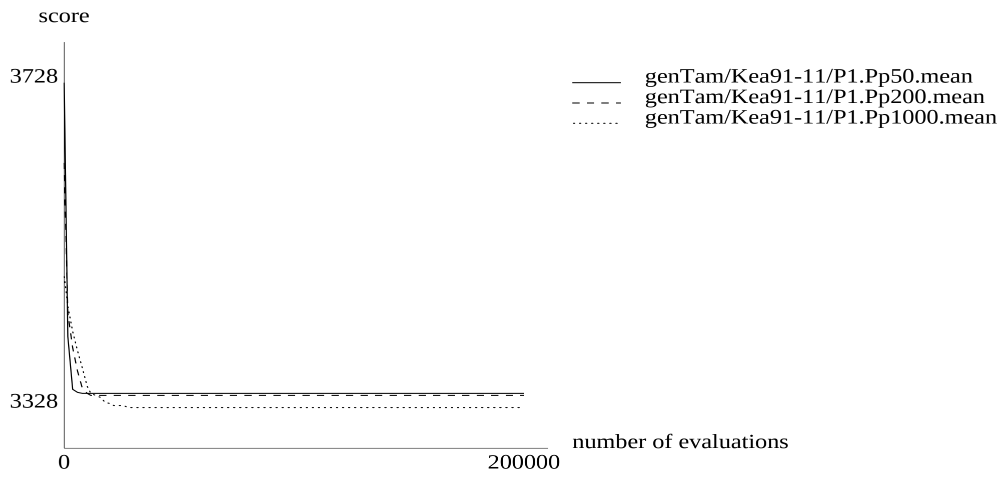
This appendix shows the state of convergence of GAs to see the effect of population size. The name of each GA is denoted as follows.

rrraaa/*fff*/P*ggg*.P*ppp*

where *rrr* =	reproduction method (gen, one, two)
aaa =	algorithm (Cea, Tam, DK, Tam2, DK2, Kad)
fff =	the name of FLP (Kea91-11, TL91-5, etc.)
ggg =	the number of populations (1, 4, 10)
ppp =	population size $\times$ the number of populations (50, 200, 1000)

The horizontal axis indicates the number of evaluations, whereas the vertical axis indicates the mean of the best individual scores. Here, smaller score is better.





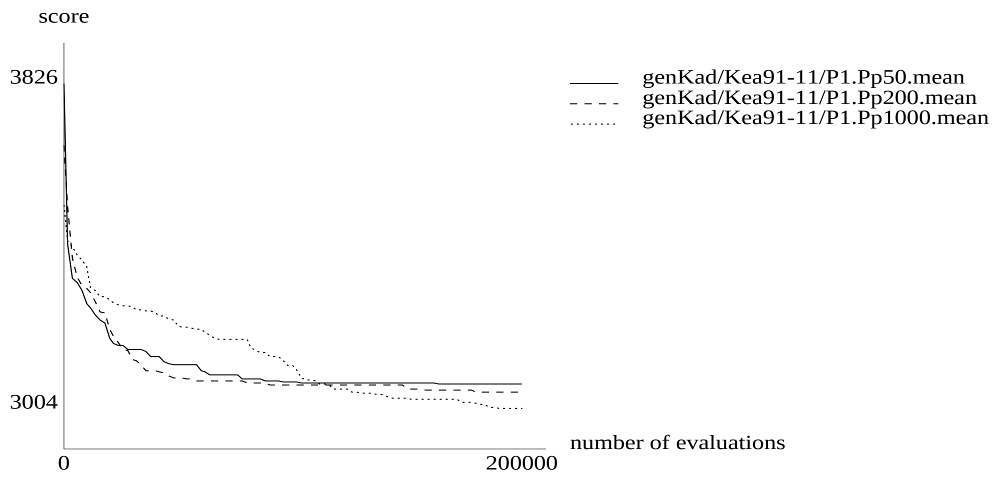


Figure D.1. A comparison of each population size in Kea91-11

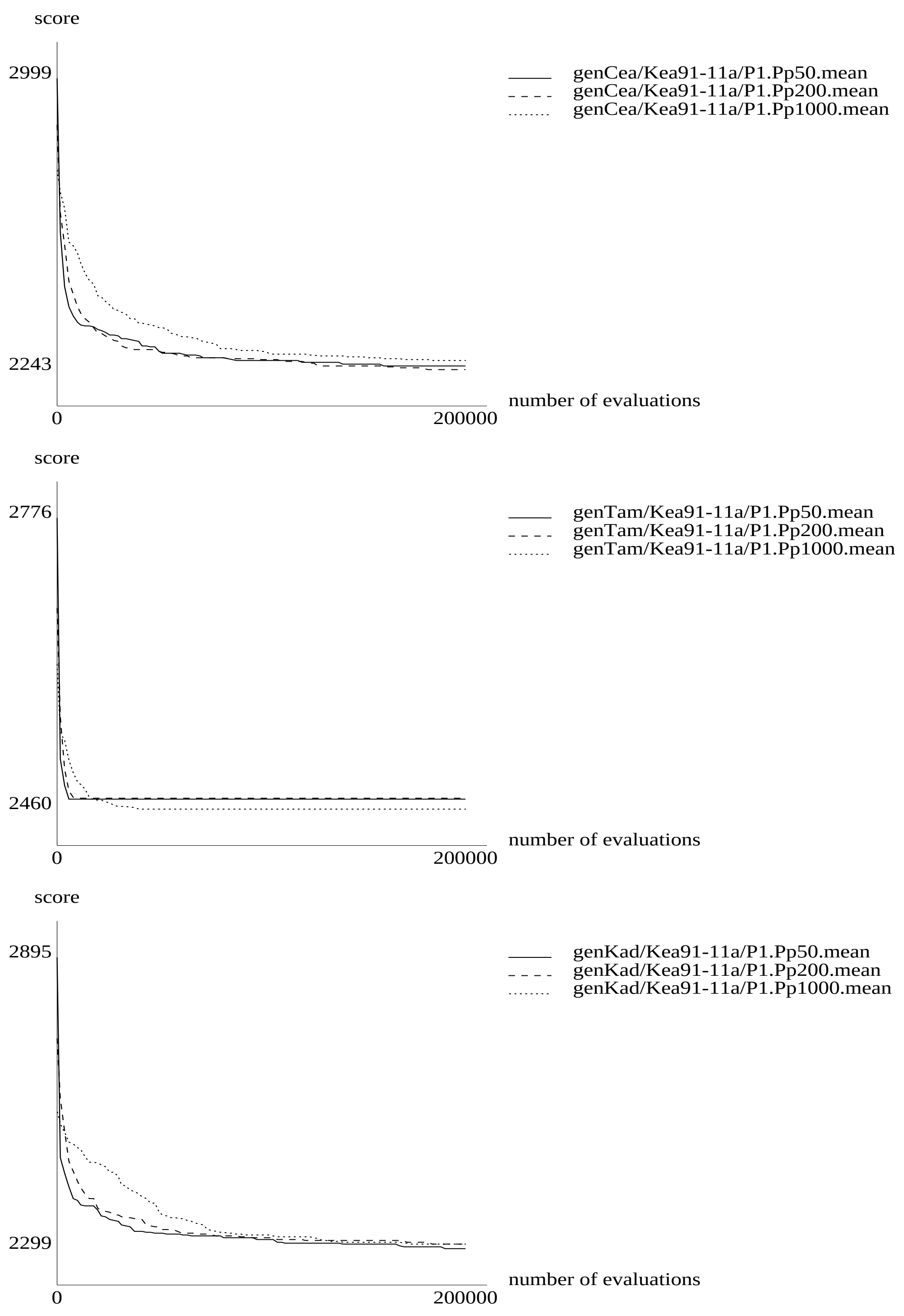
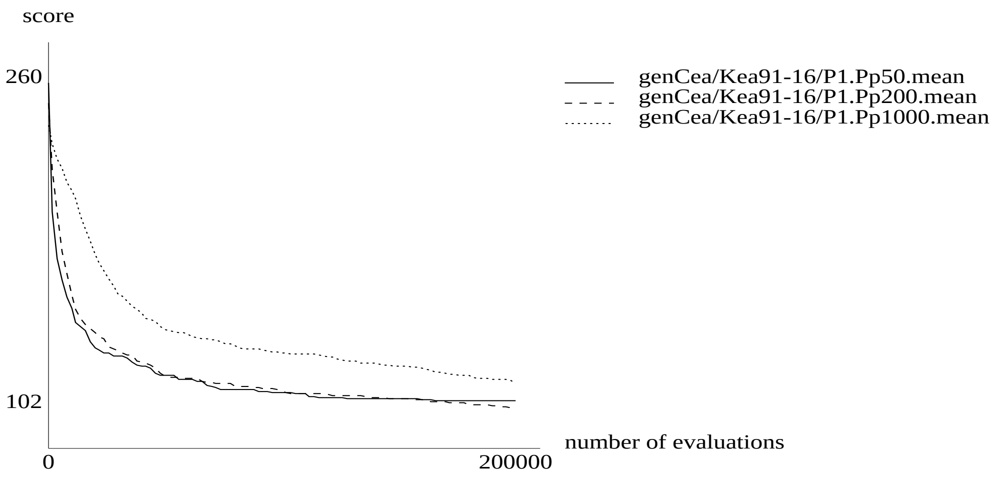
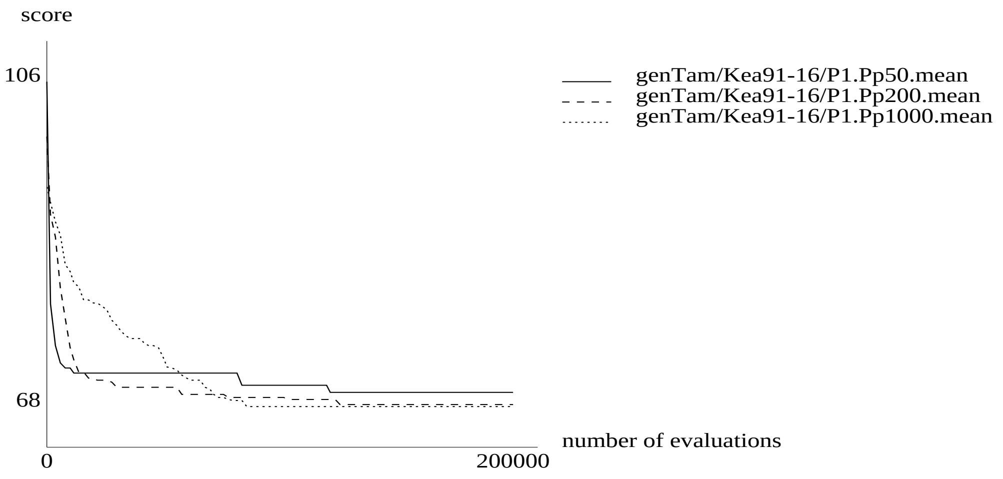


Figure D.2. A comparison of each population size in Kea91-11a





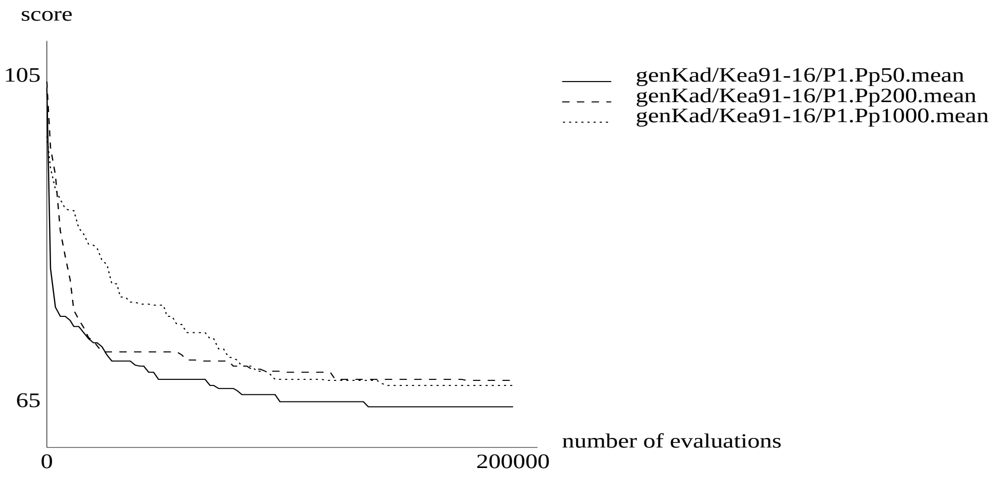
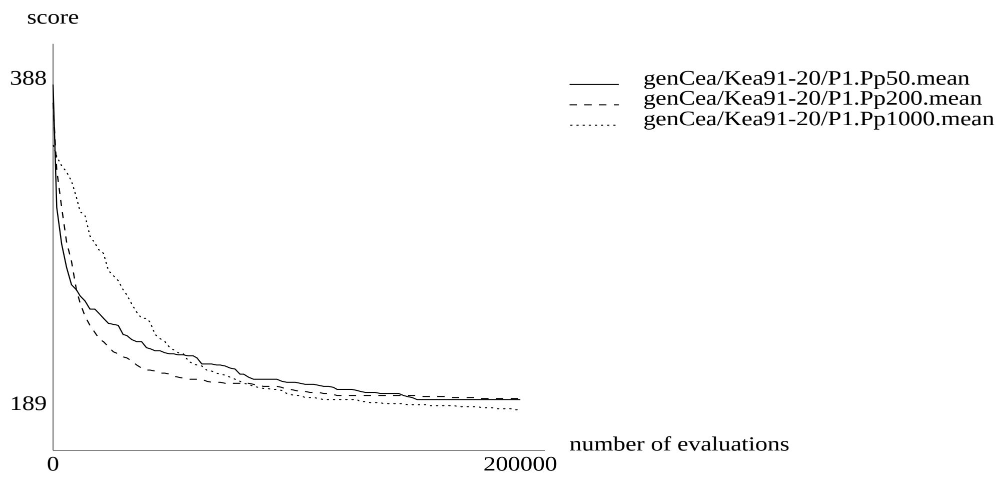
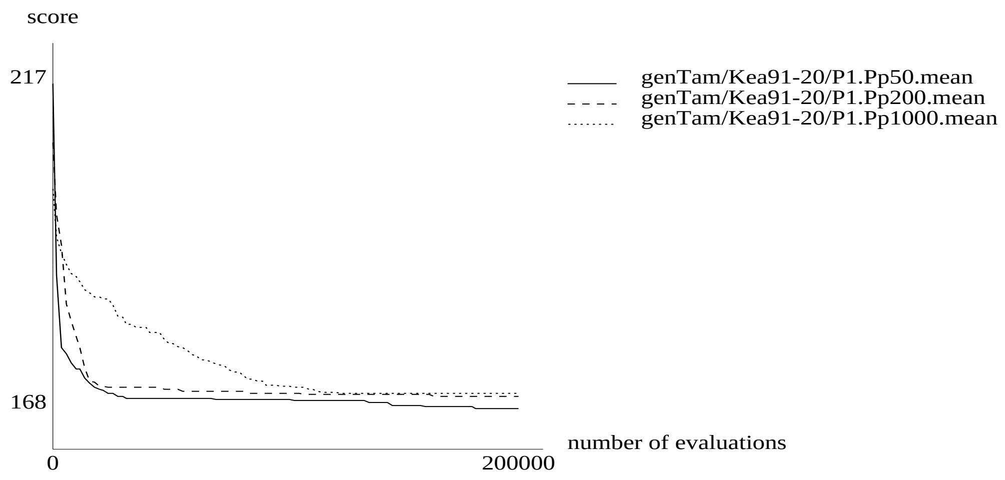


Figure D.3. A comparison of each population size in Kea91-16





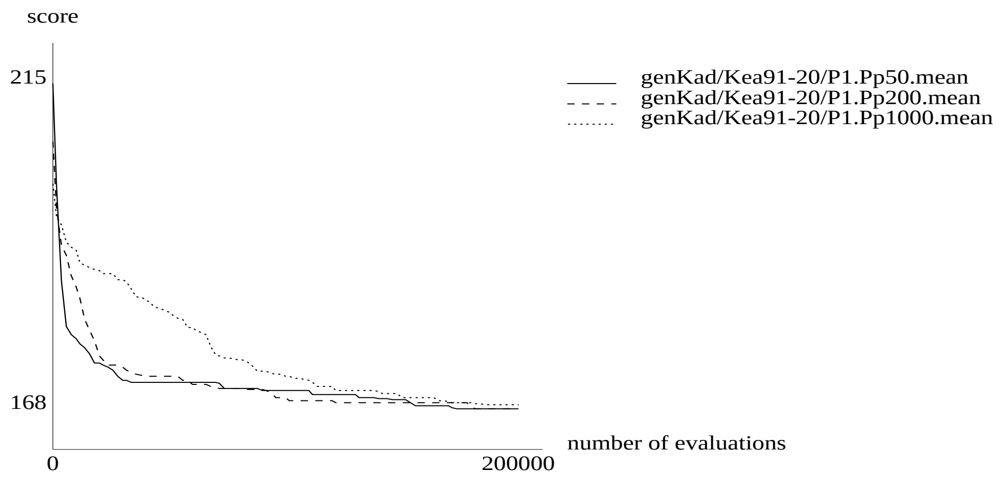
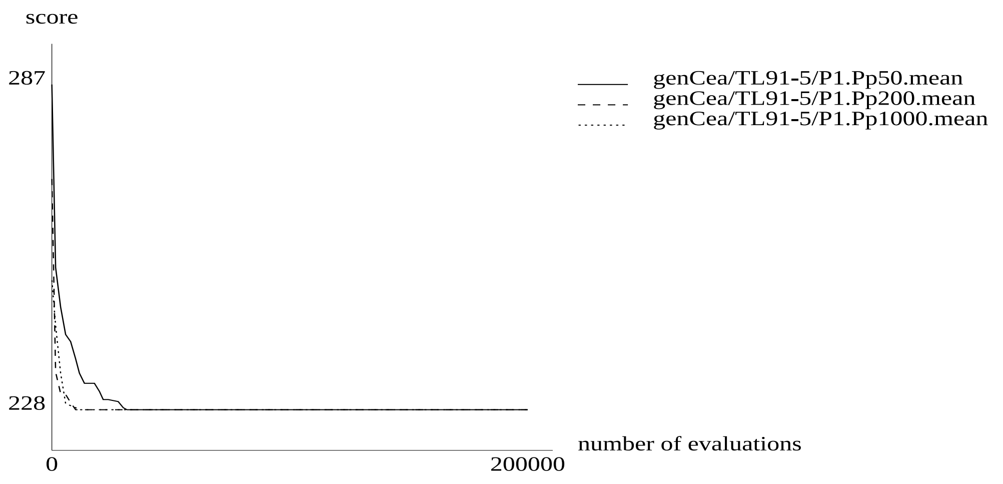
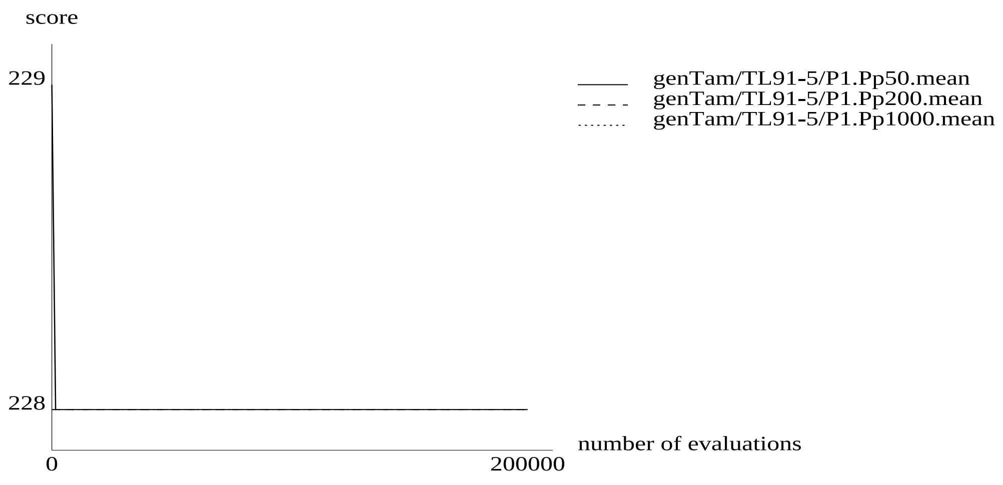


Figure D.4. A comparison of each population size in Kea91-20





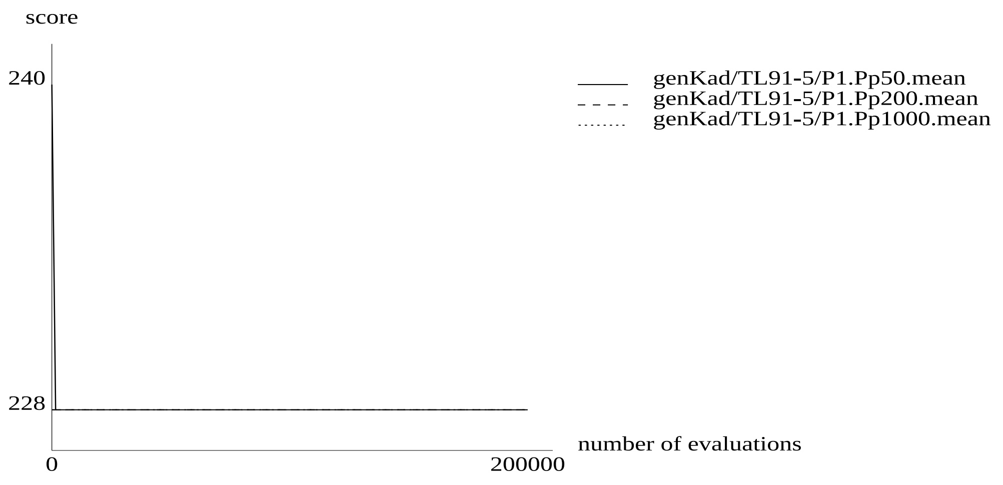
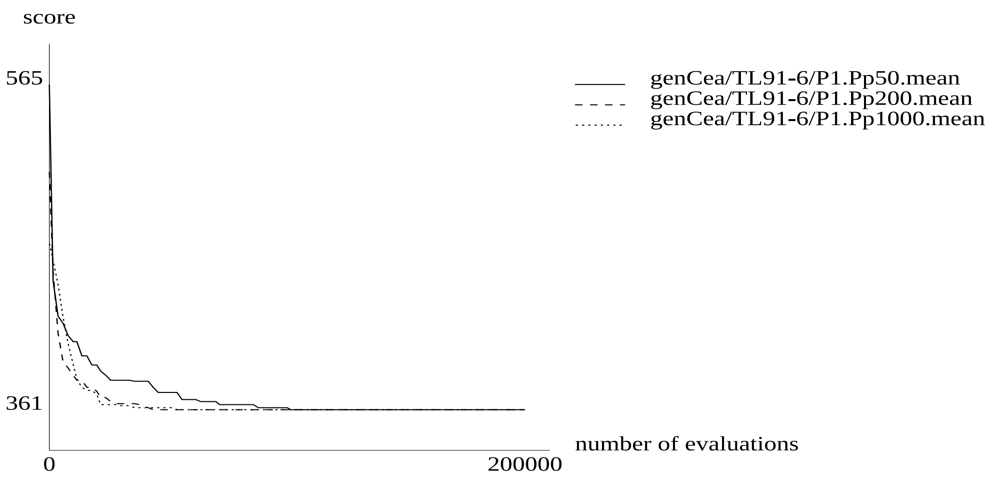
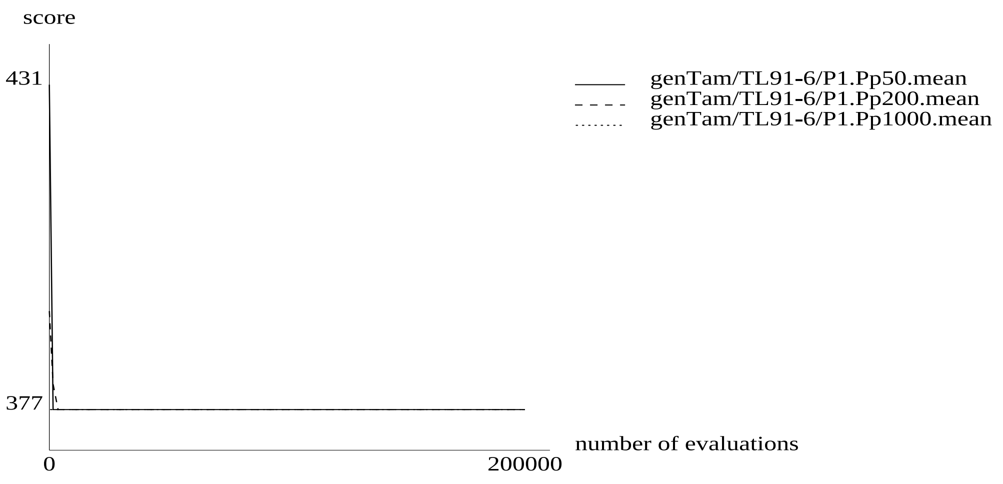


Figure D.5. A comparison of each population size in TL91-5





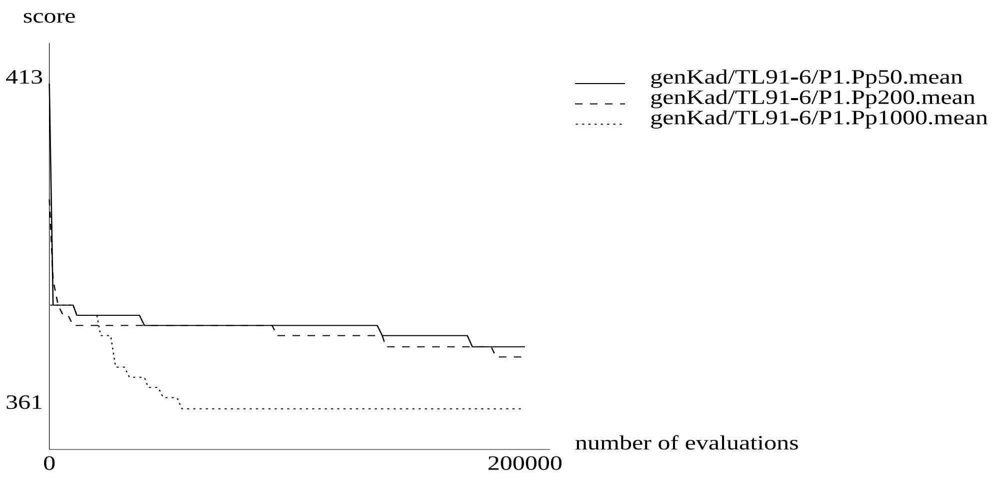
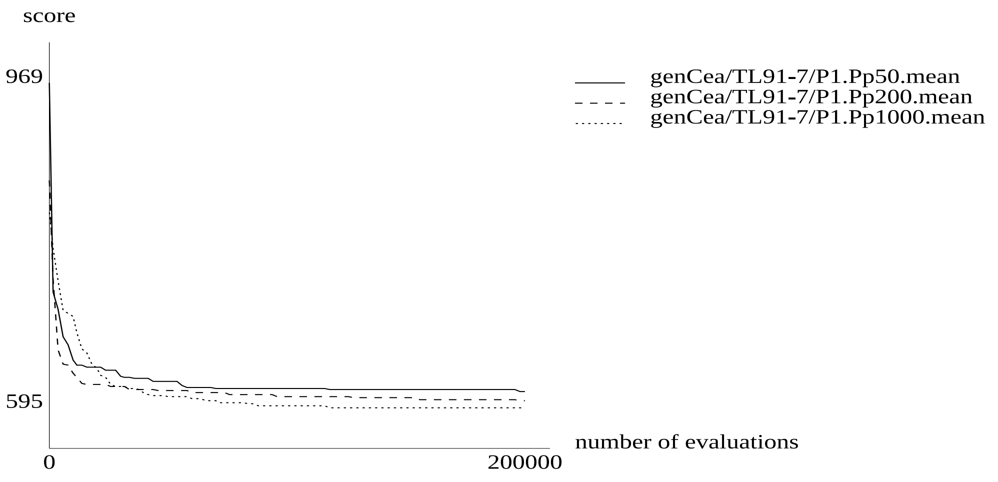
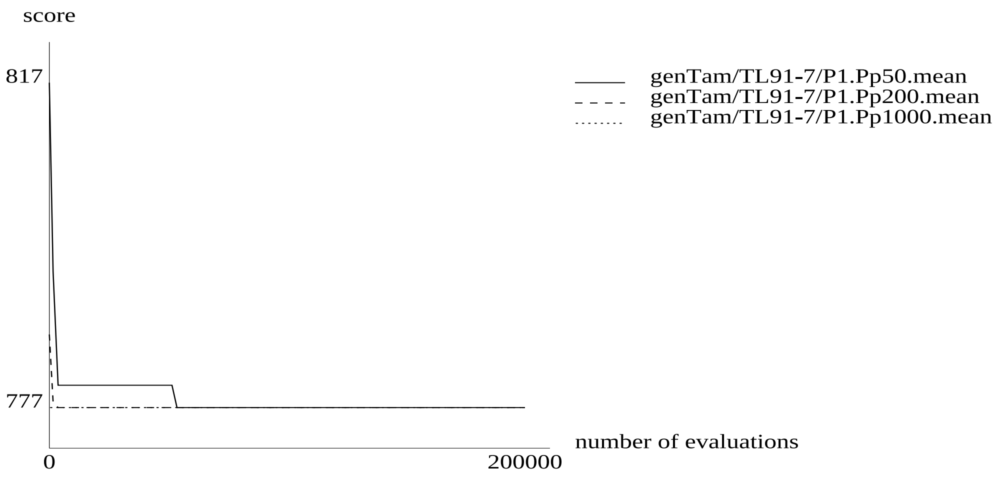


Figure D.6. A comparison of each population size in TL91-6





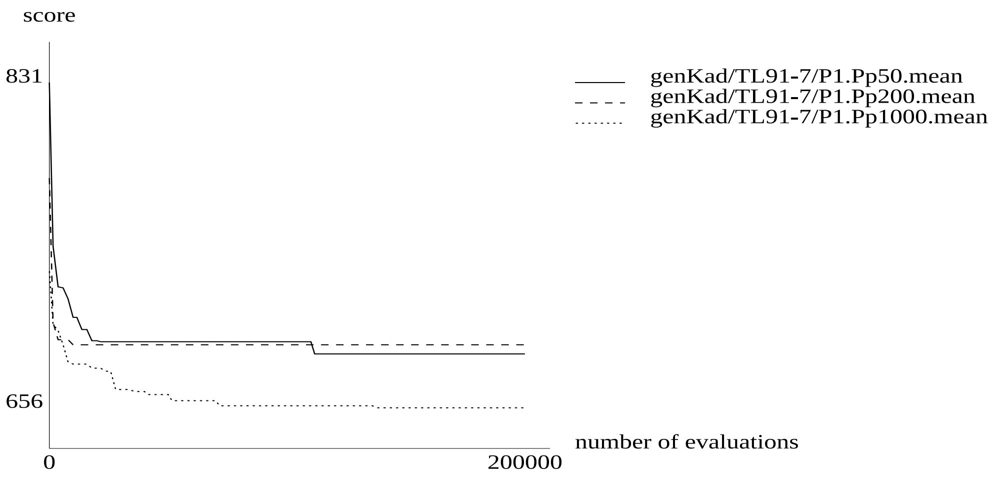
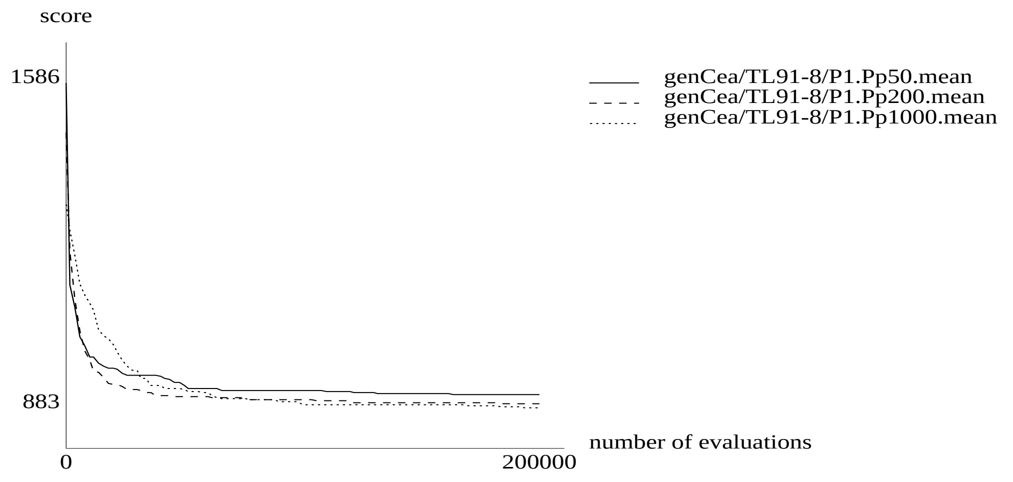
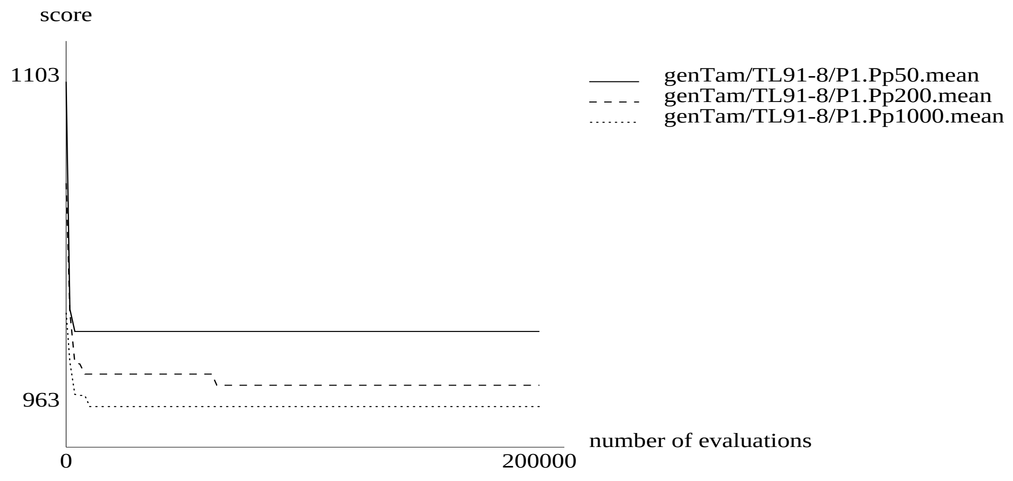


Figure D.7. A comparison of each population size in TL91-7





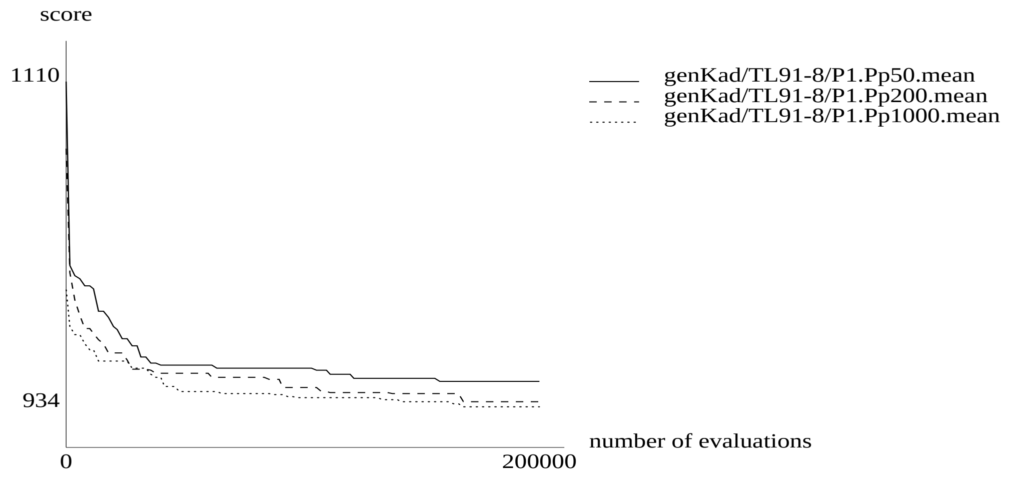
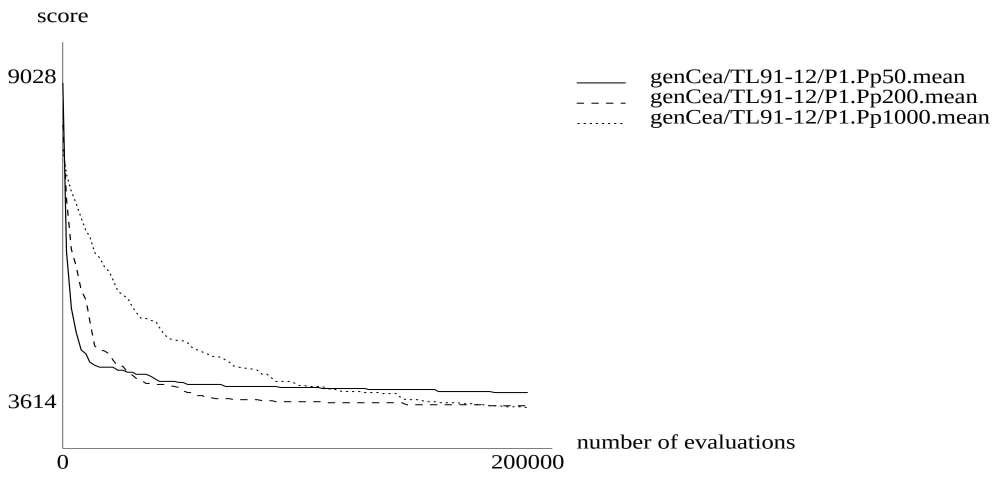
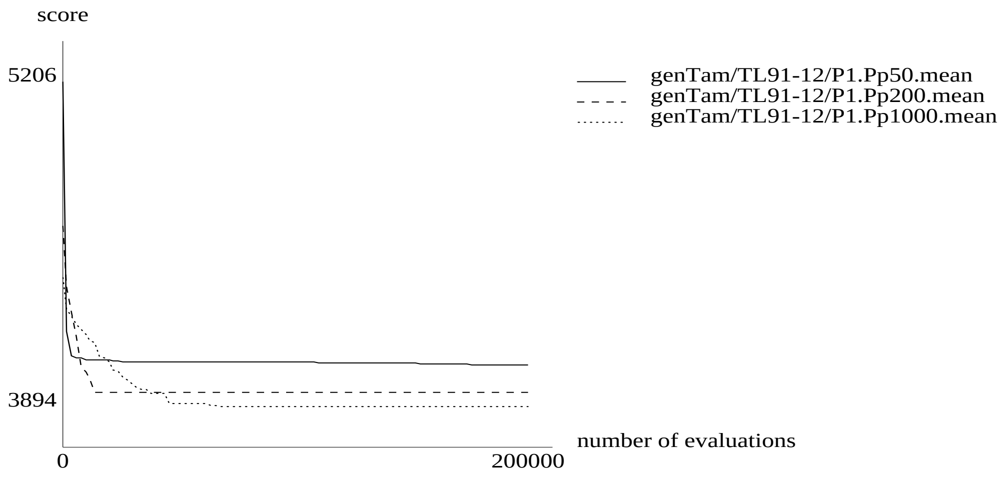


Figure D.8. A comparison of each population size in TL91-8





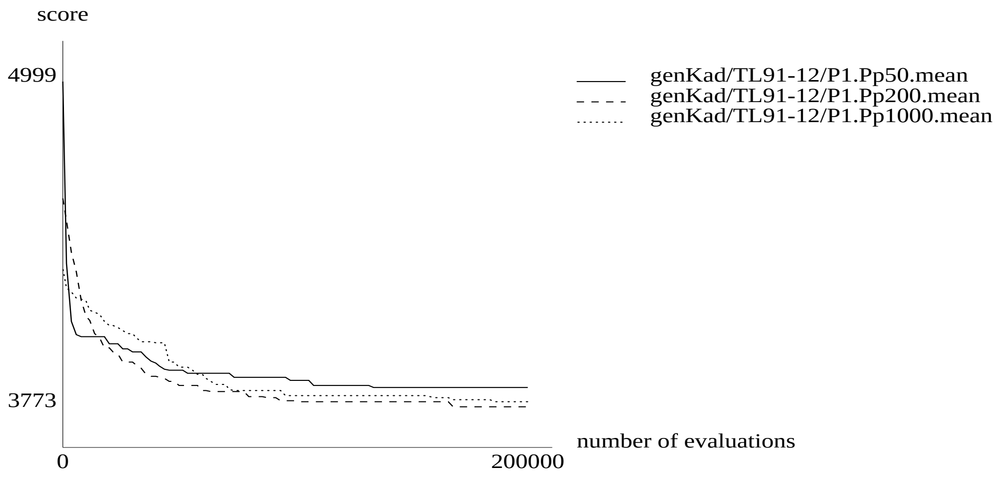
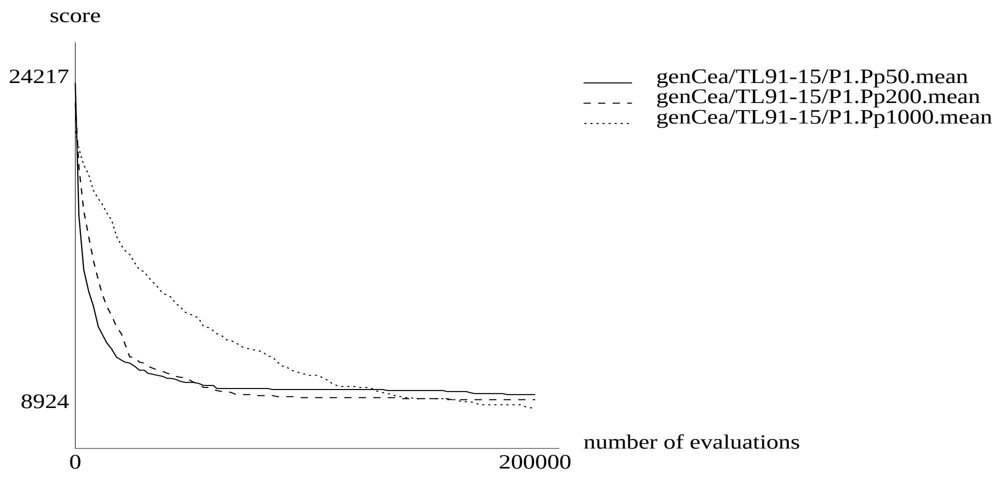
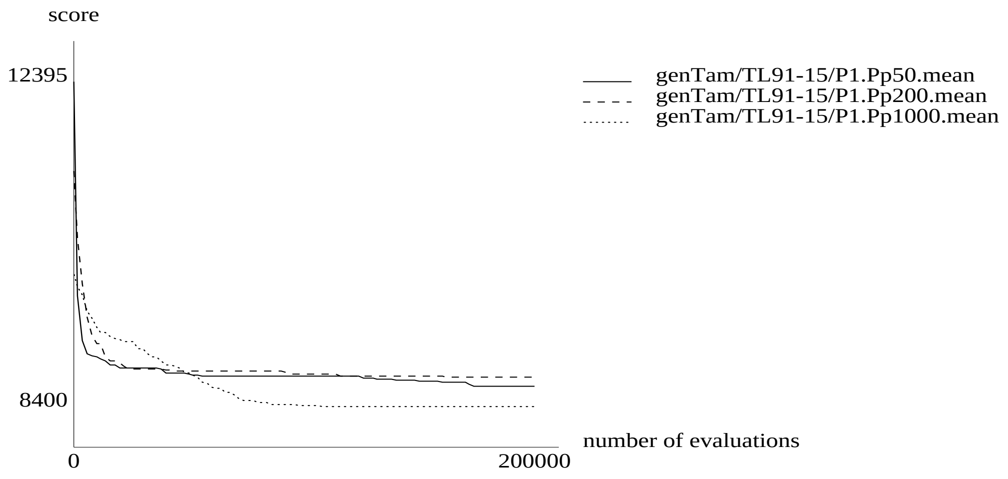


Figure D.9. A comparison of each population size in TL91-12





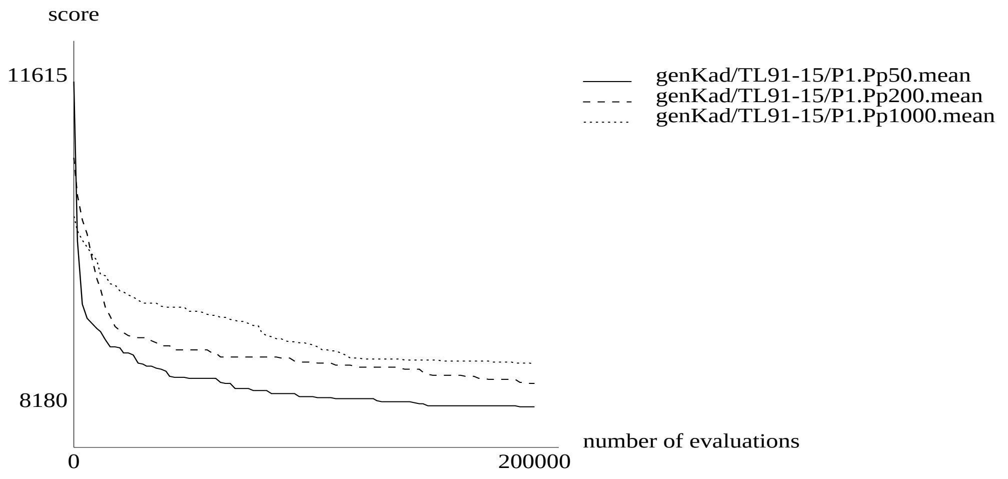
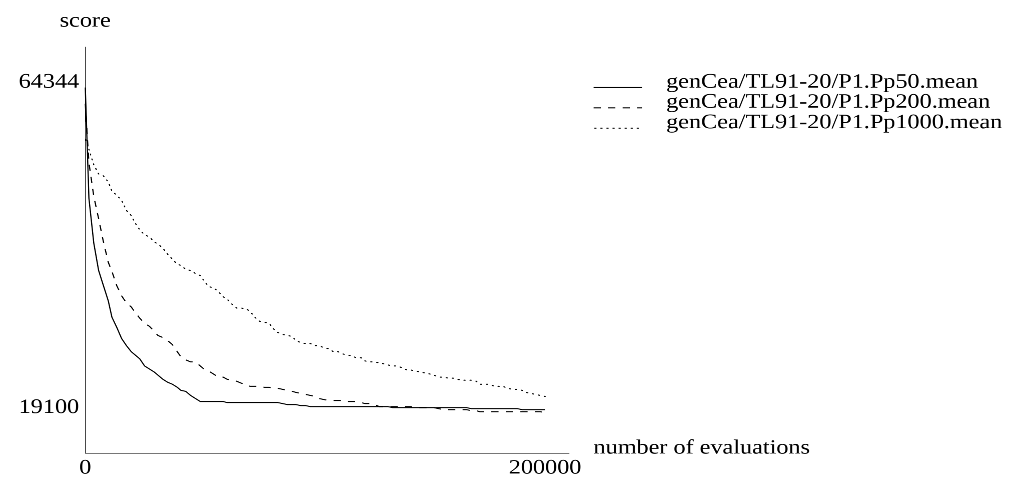
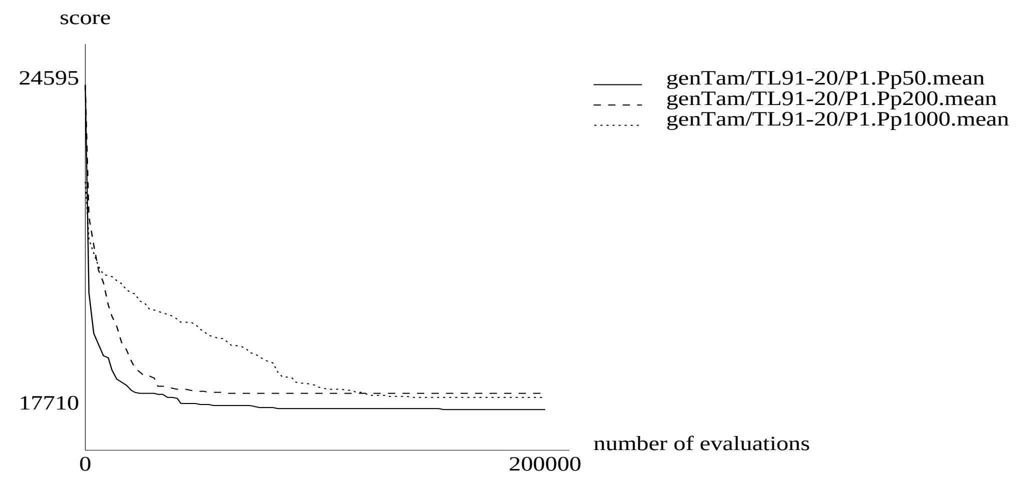


Figure D.10. A comparison of each population size in TL91-15





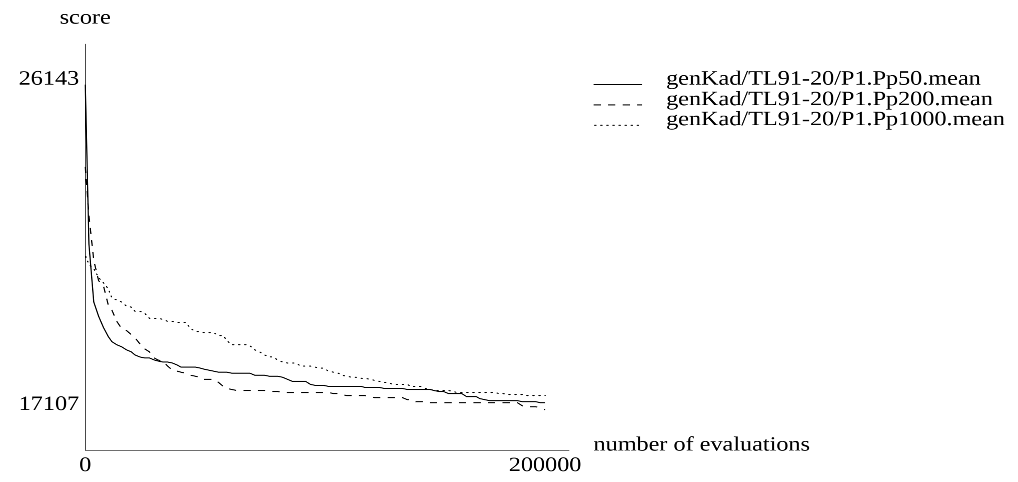
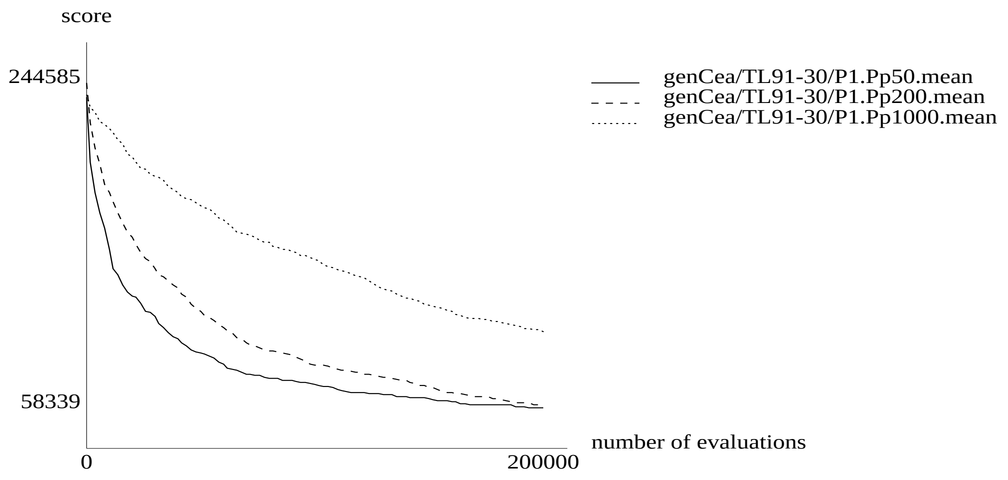
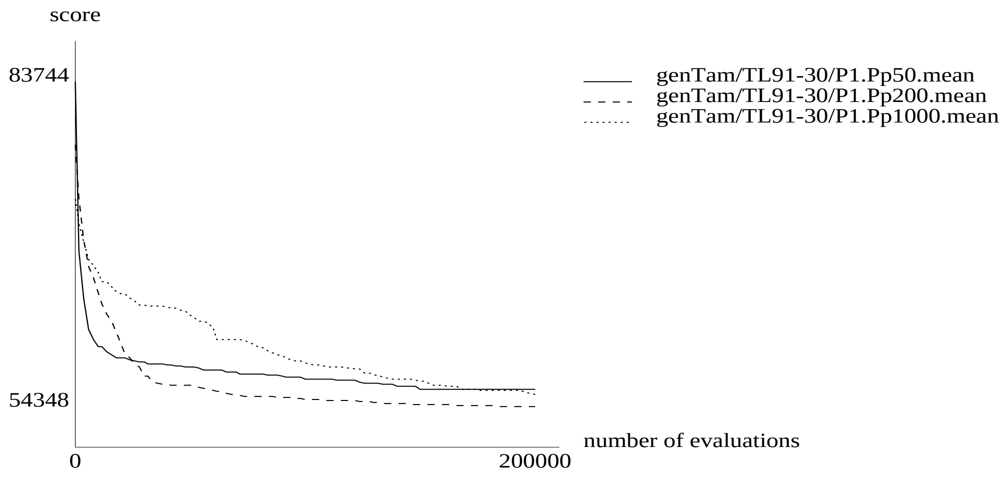


Figure D.11. A comparison of each population size in TL91-20





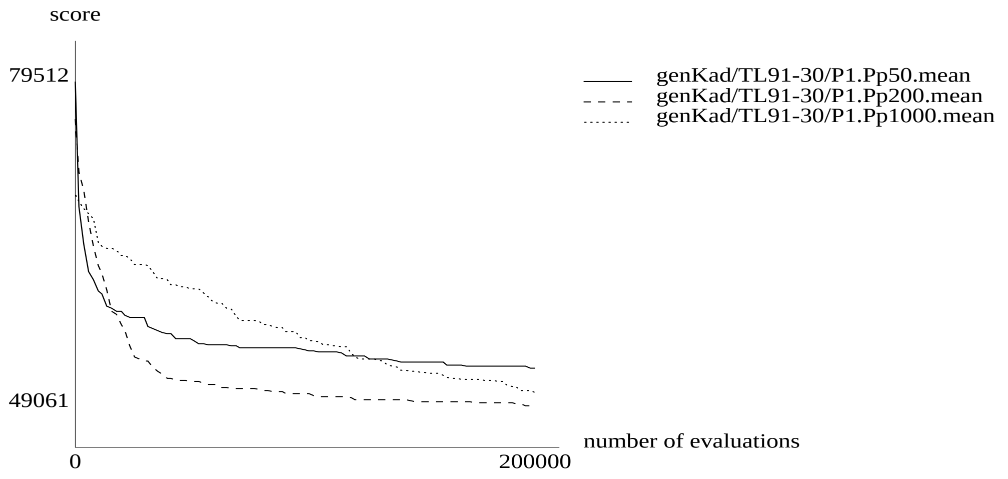
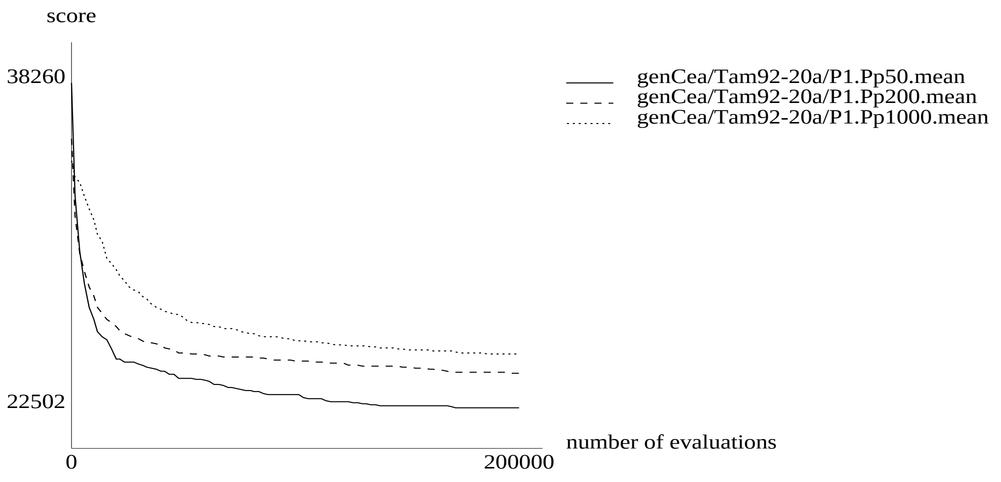
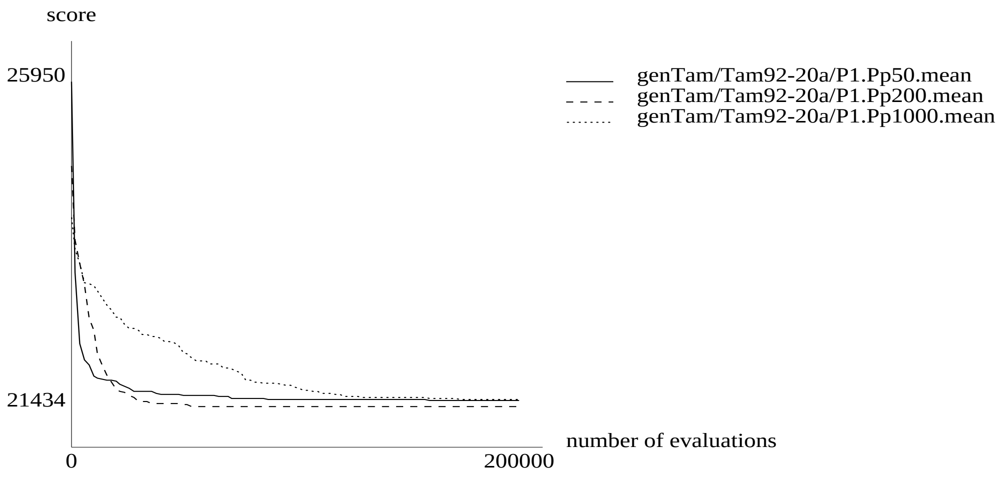


Figure D.12. A comparison of each population size in TL91-30





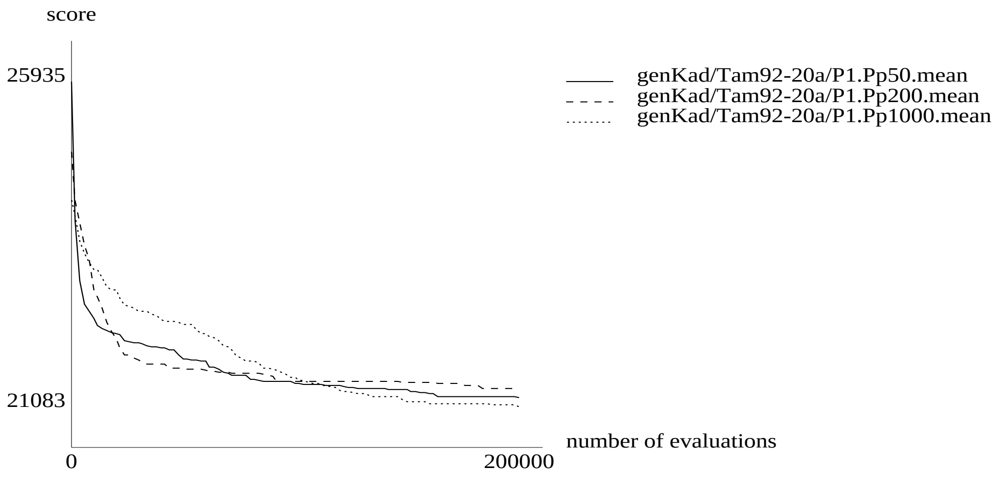
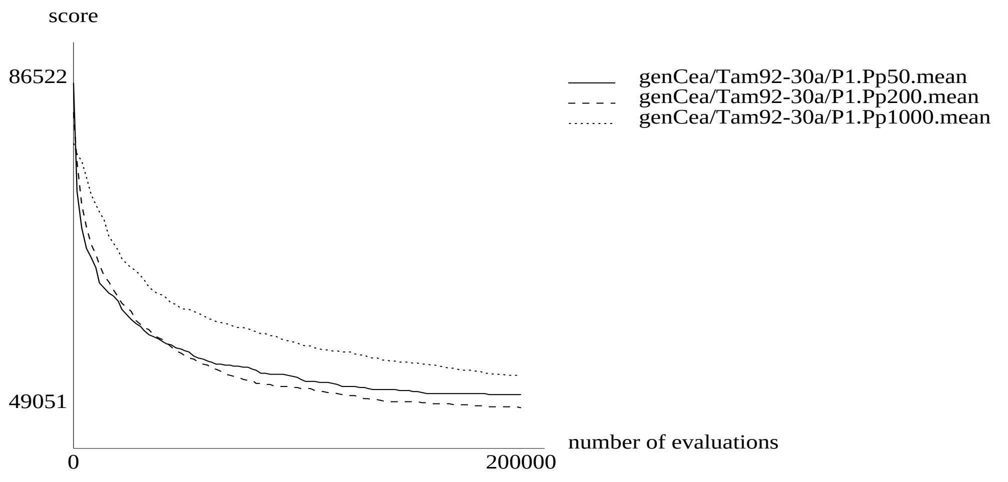
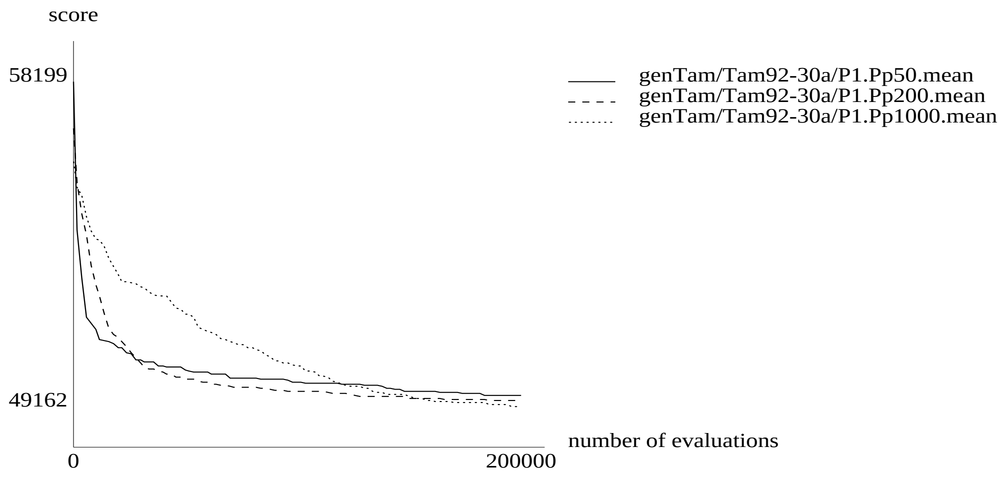


Figure D.13. A comparison of each population size in Tam92-20a





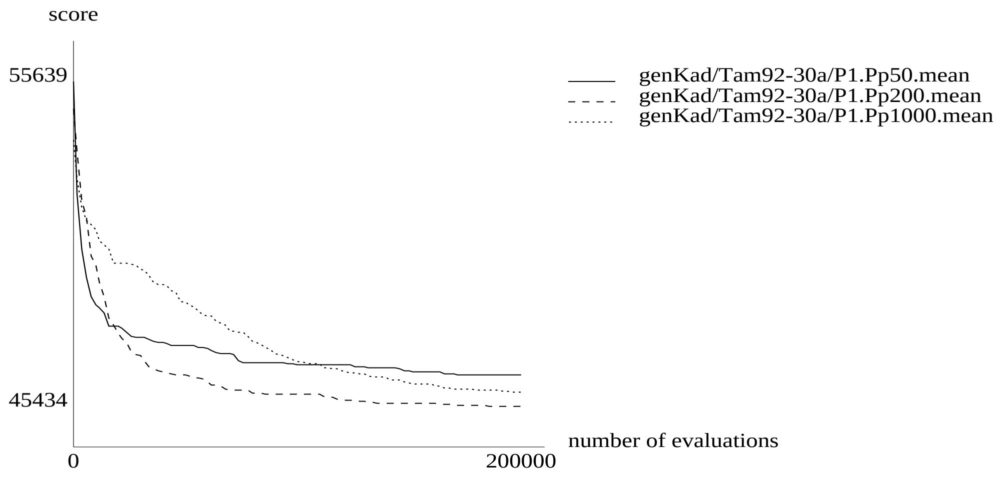
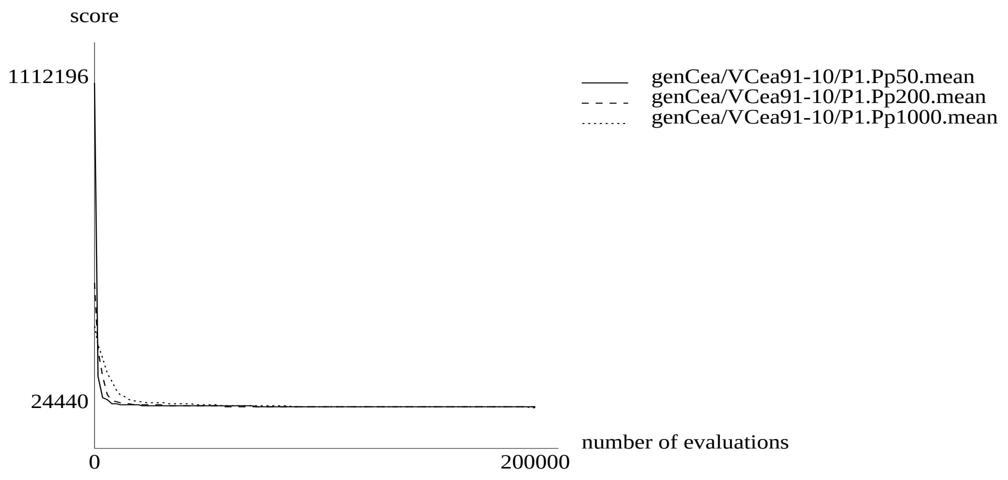
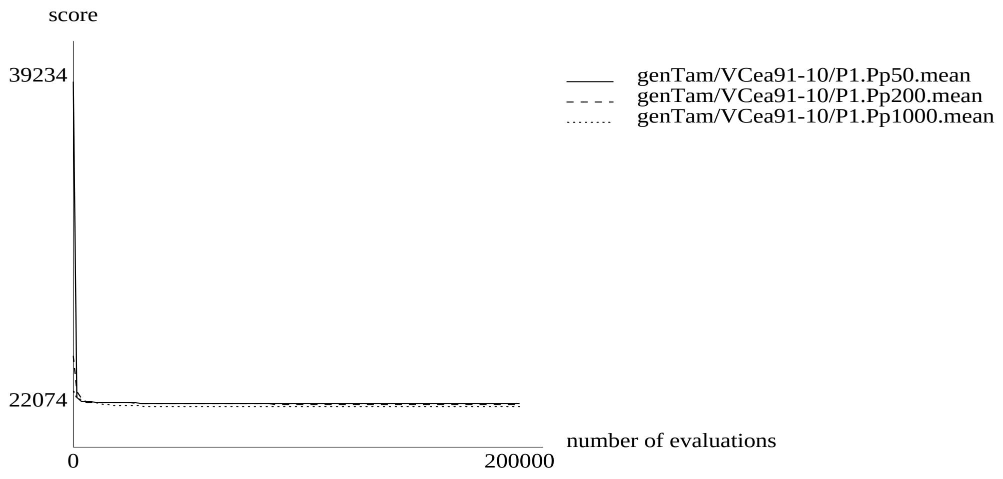


Figure D.14. A comparison of each population size in Tam92-30a





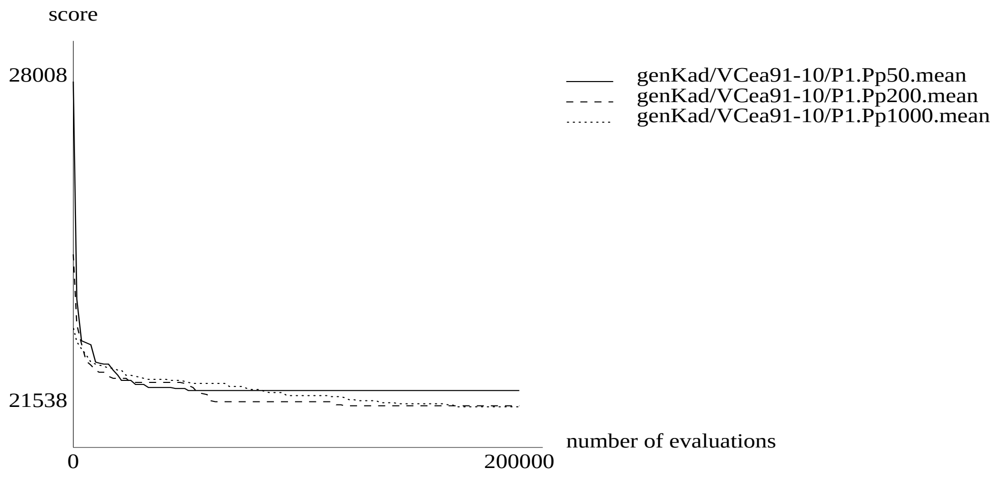


Figure D.15. A comparison of each population size in VCea91-10

Appendix E

Investigation Results for the Number of Populations

This appendix shows the state of convergence of GAs to see the effect of number of populations. The name of each GA is denoted as follows.

rrraaa/*fff*/P*ggg*.Pp*ppp*

where *rrr*	=	reproduction method (gen, one, two)
aaa	=	algorithm (Cea, Tam, DK, Tam2, DK2, Kad)
fff	=	the name of FLP (Kea91-11, TL91-5, etc.)
ggg	=	the number of populations (1, 4, 10)
ppp	=	population size $\times$ the number of populations (50, 200, 1000)

The horizontal axis indicates the number of evaluations, whereas the vertical axis indicates the mean of the best individual scores. Here, smaller score is better.

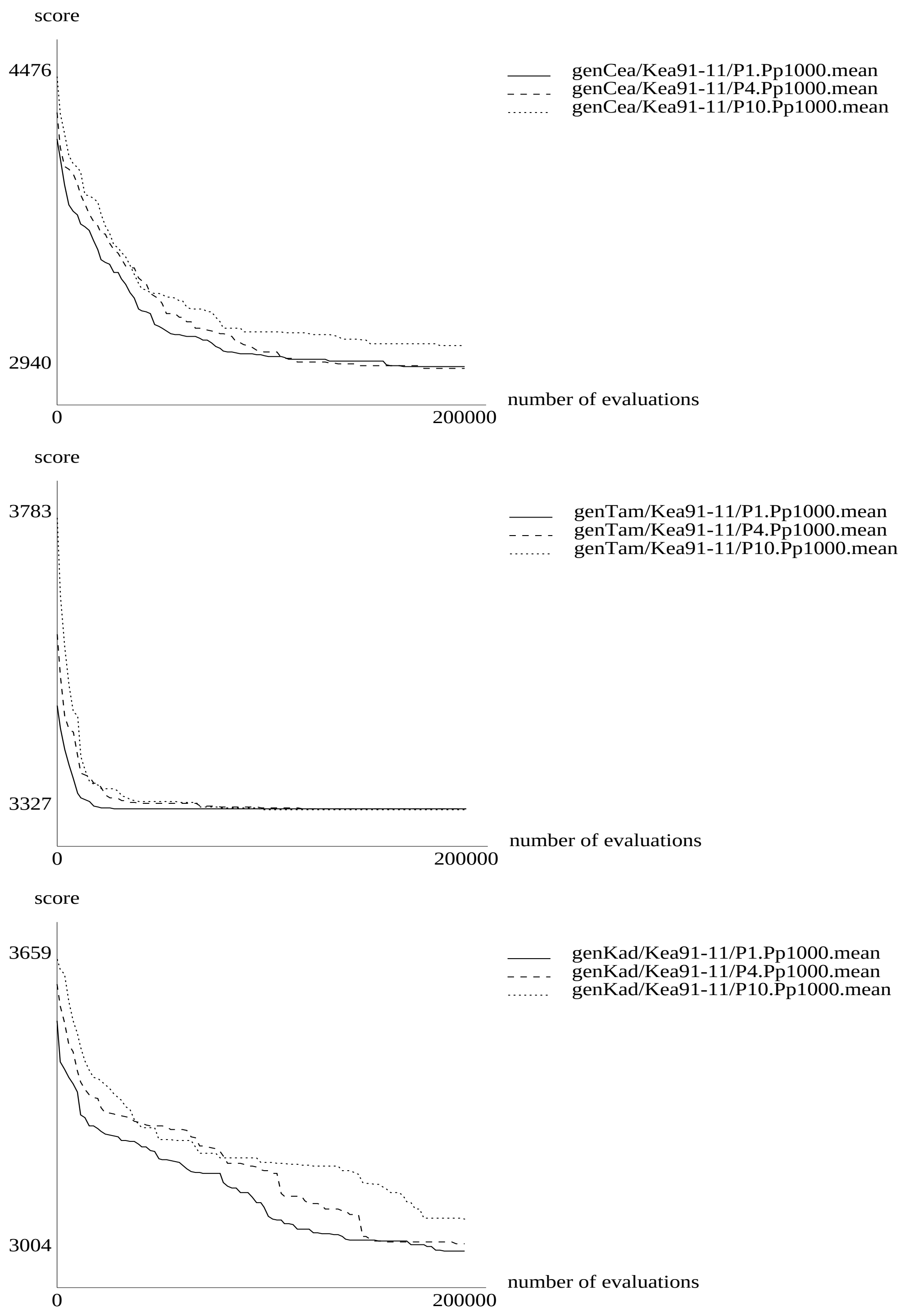
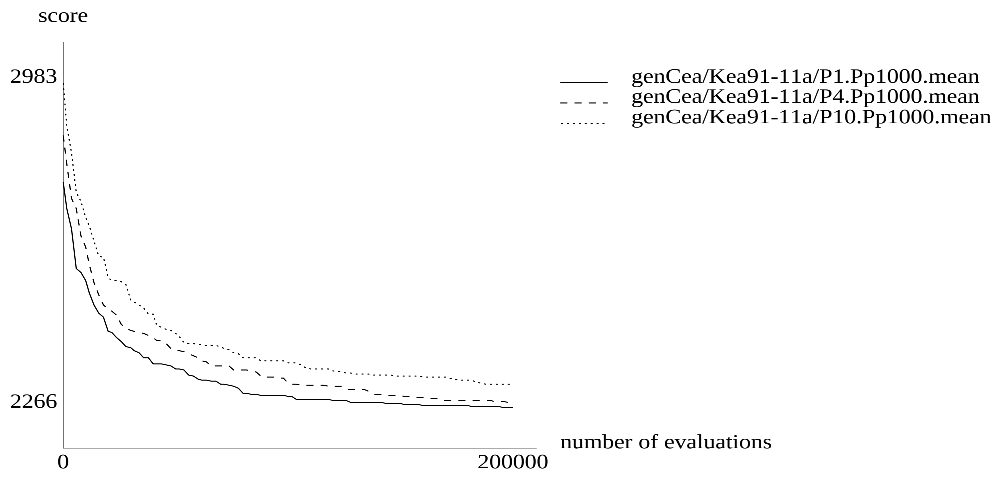
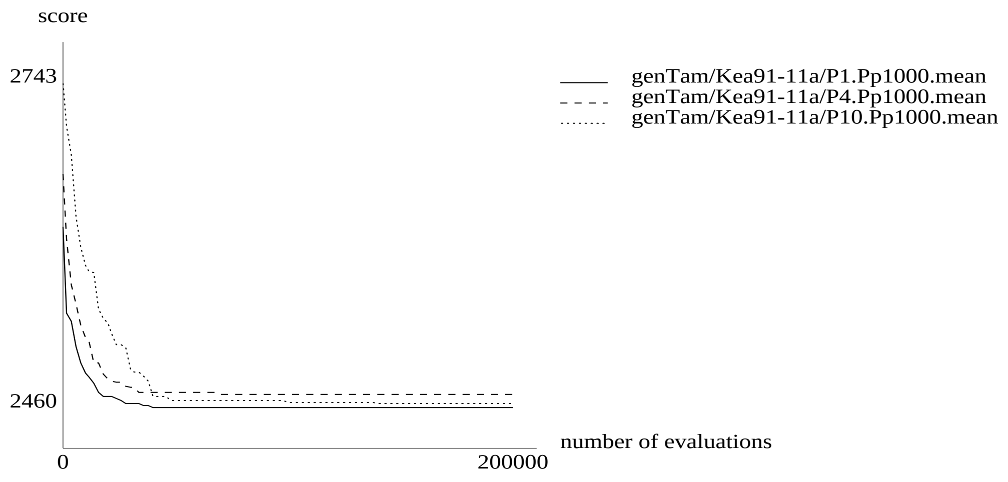


Figure E.1. A comparison of each population number in Kea91-11





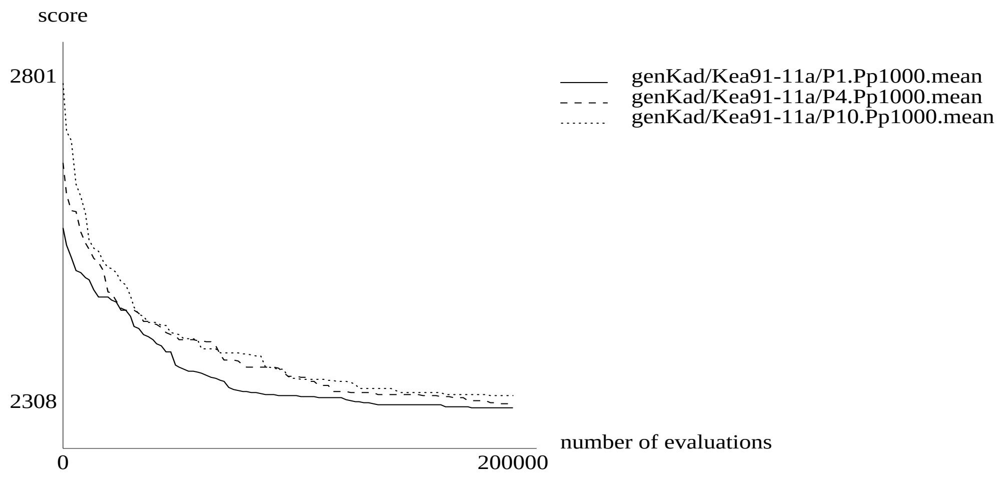
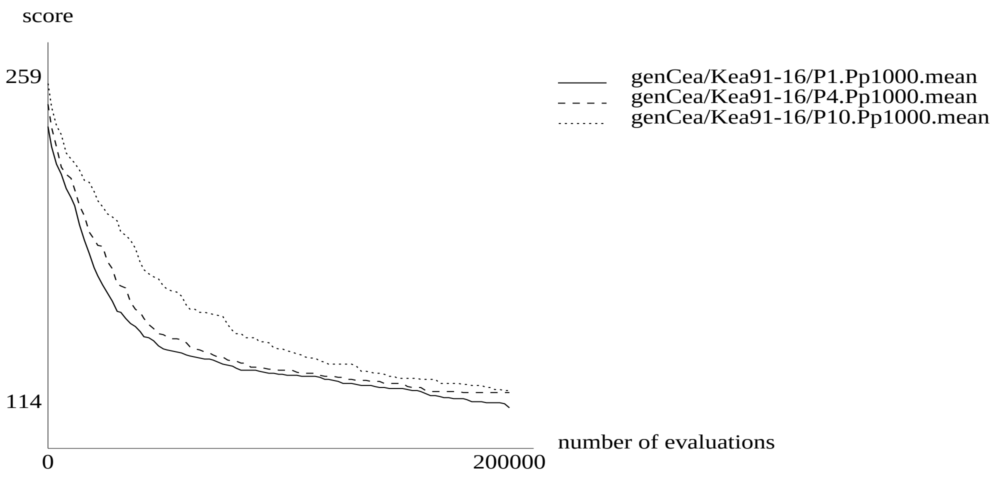
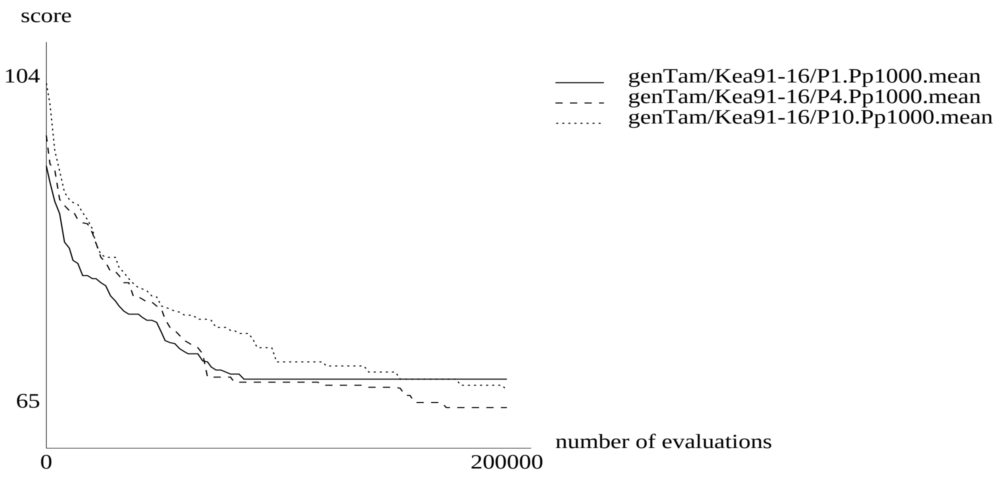


Figure E.2. A comparison of each population number in Kea91-11a





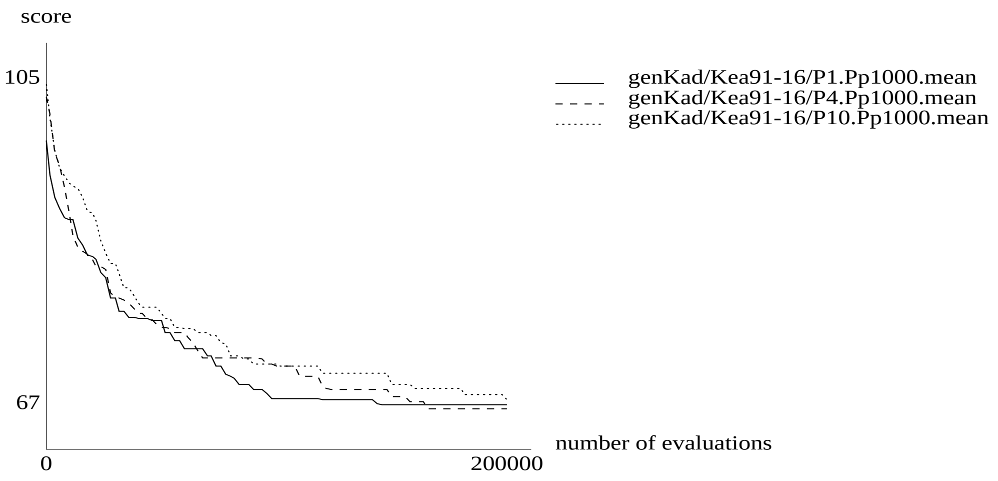
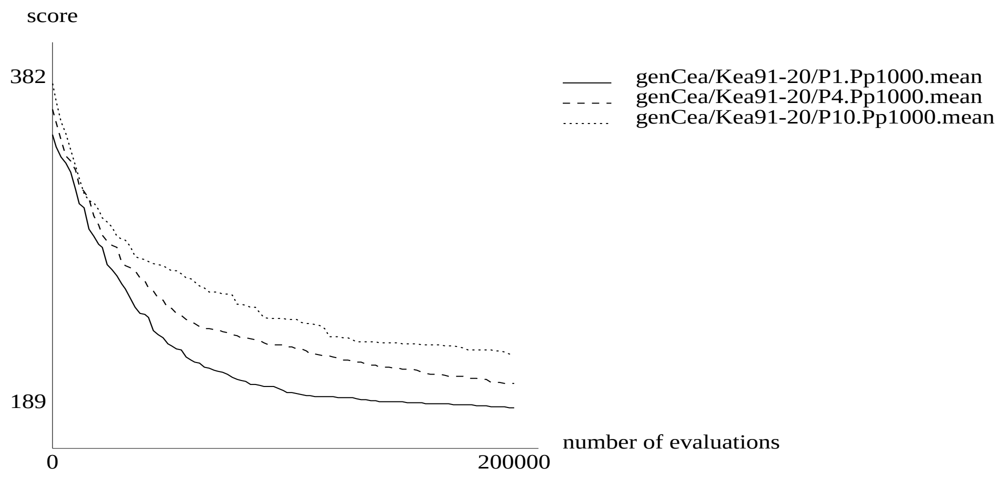
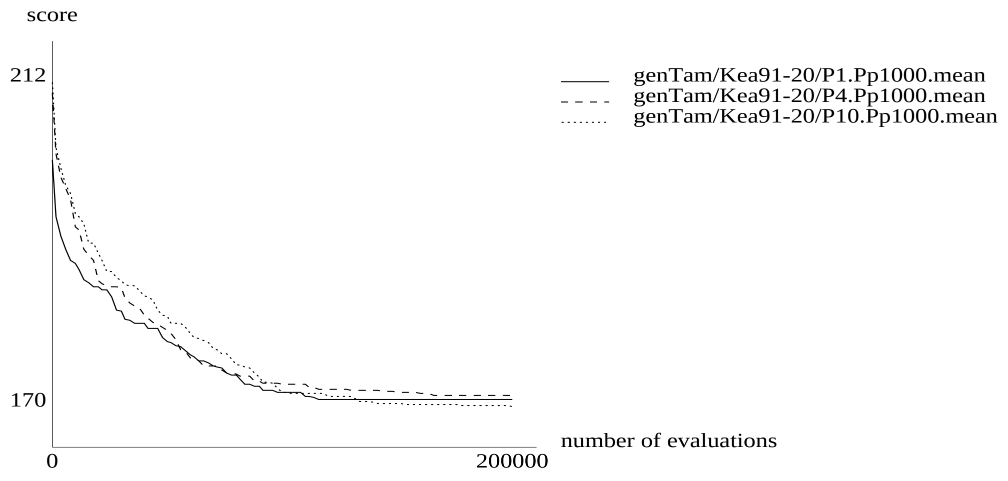


Figure E.3. A comparison of each population number in Kea91-16





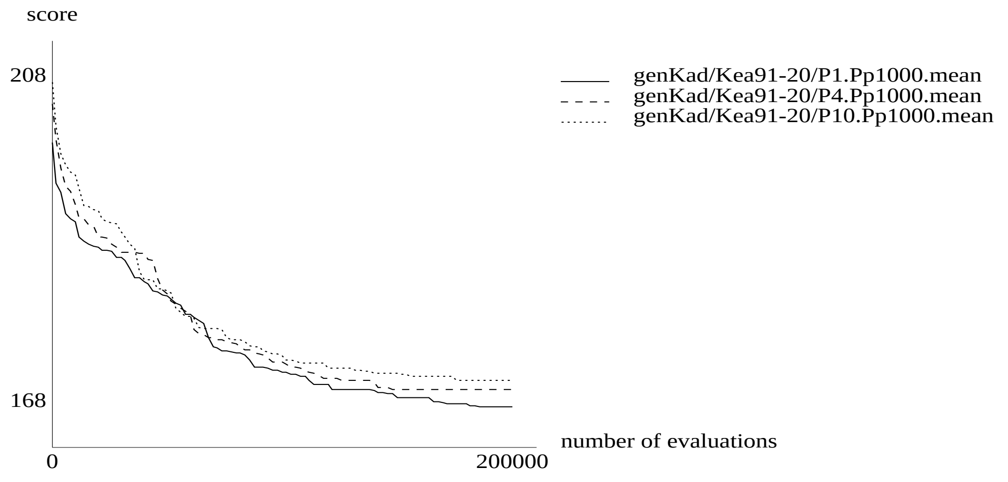
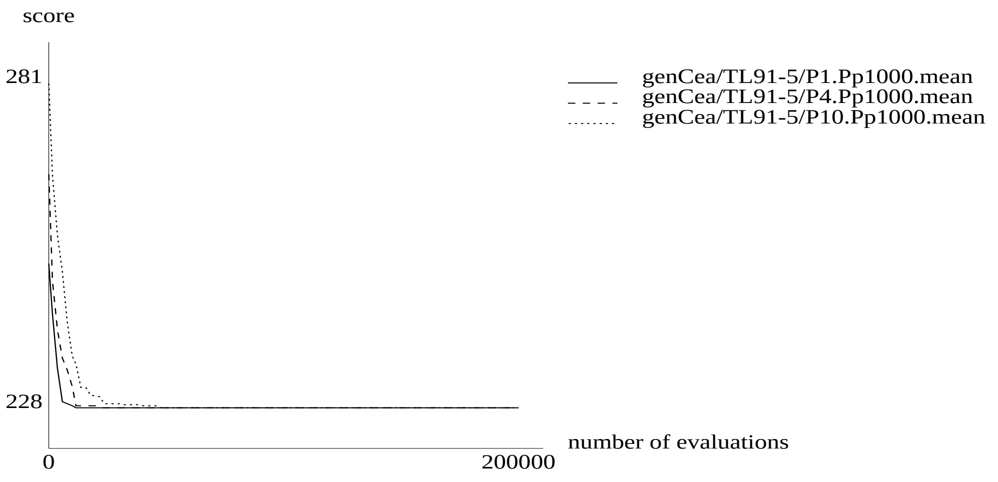
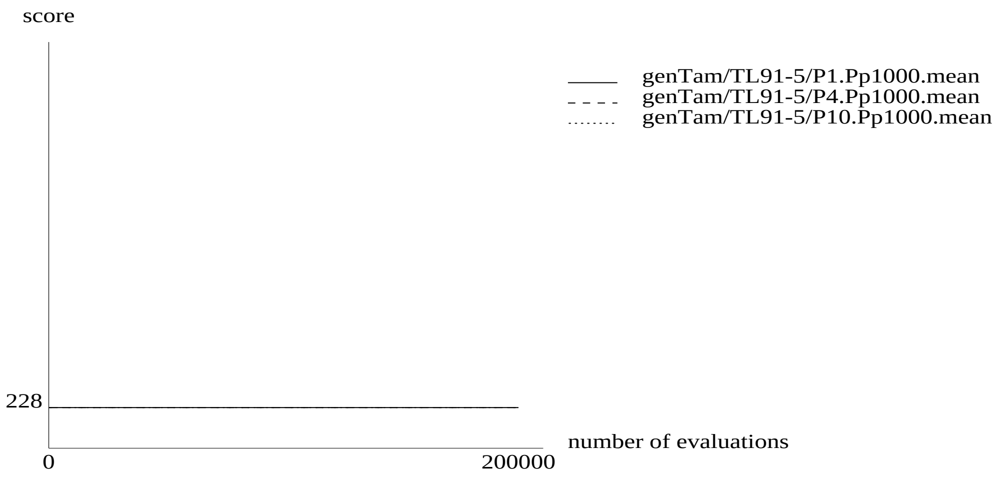


Figure E.4. A comparison of each population number in Kea91-20





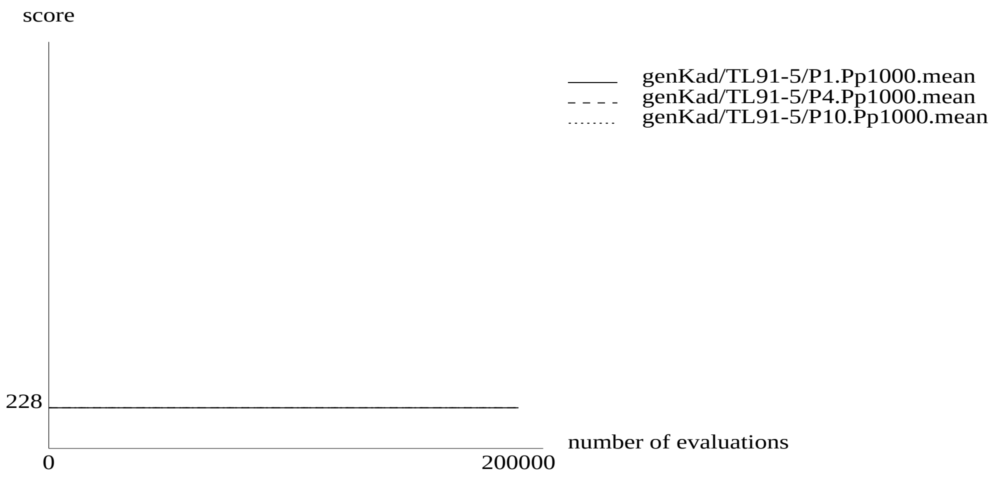
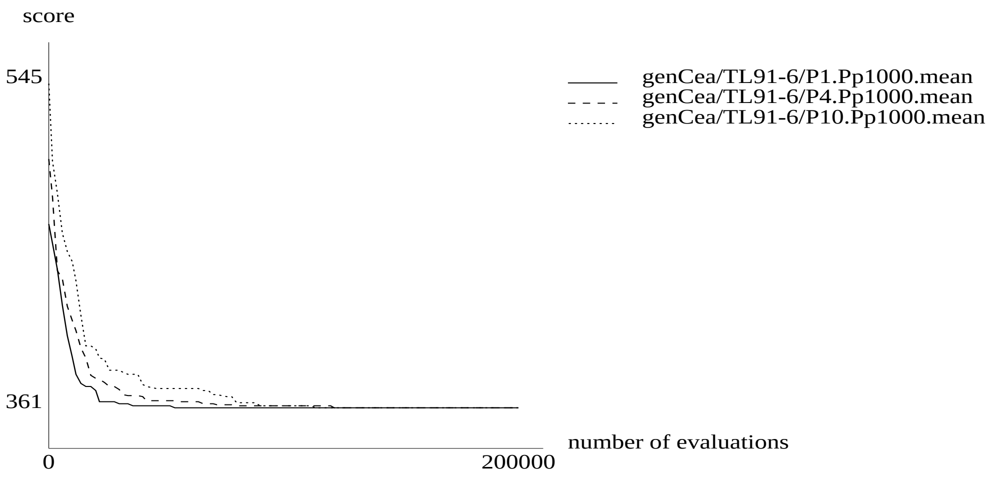
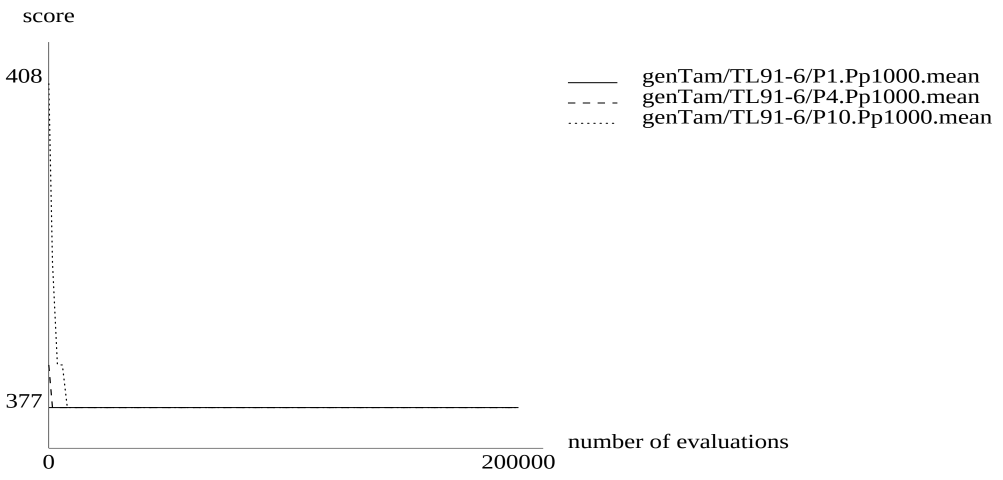


Figure E.5. A comparison of each population number in TL91-5





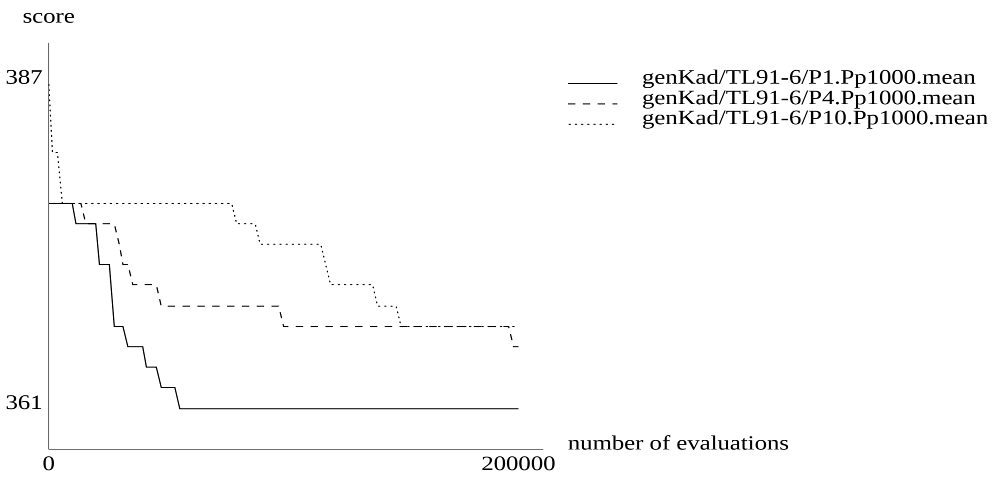
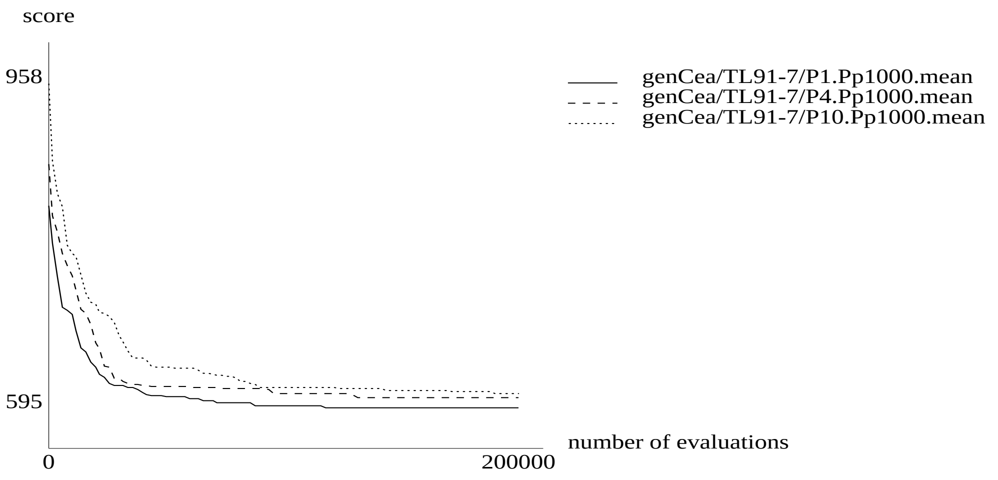
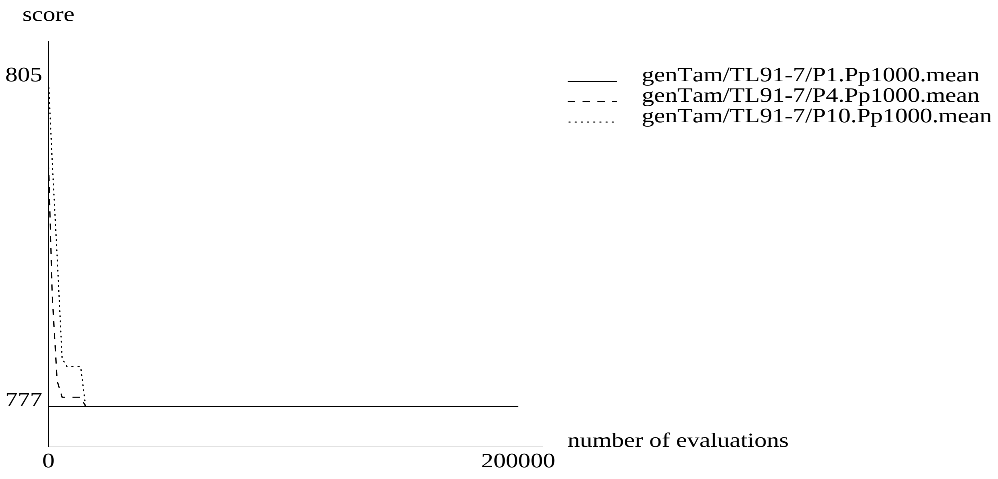


Figure E.6. A comparison of each population number in TL91-6





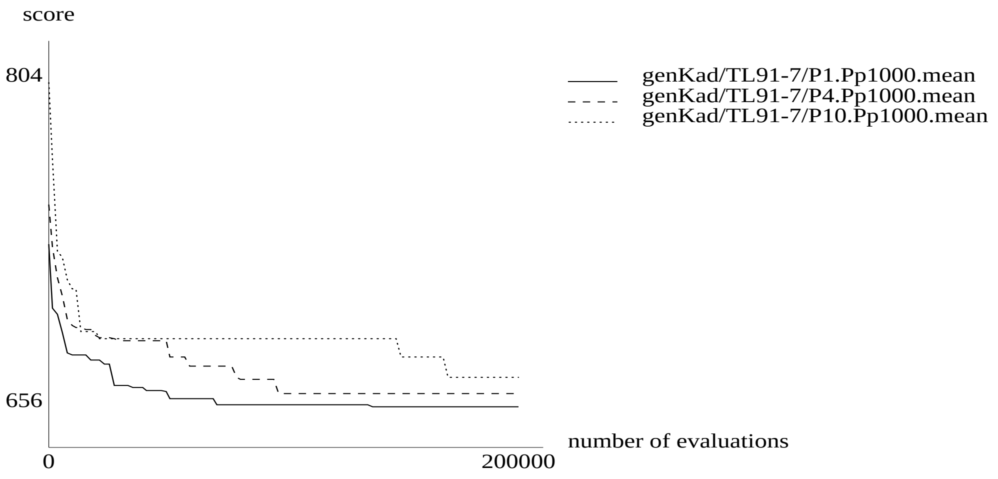
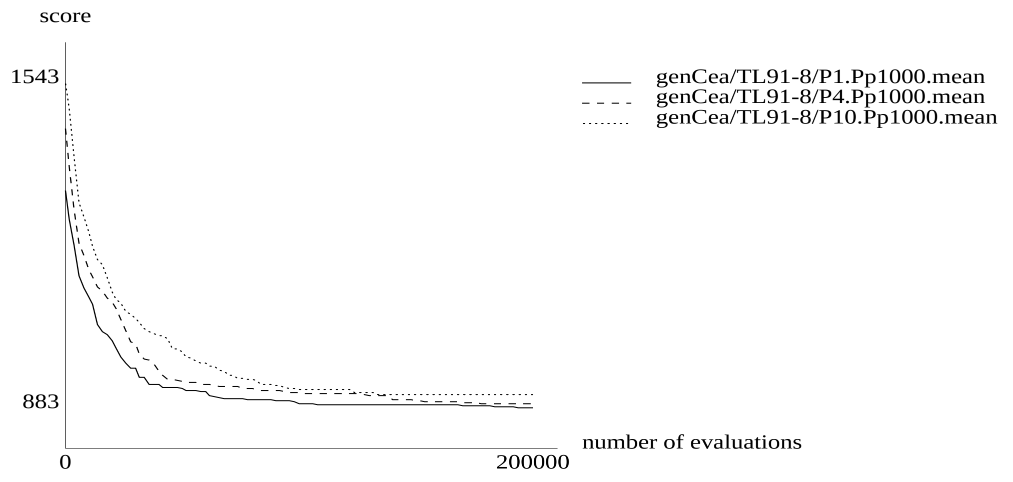
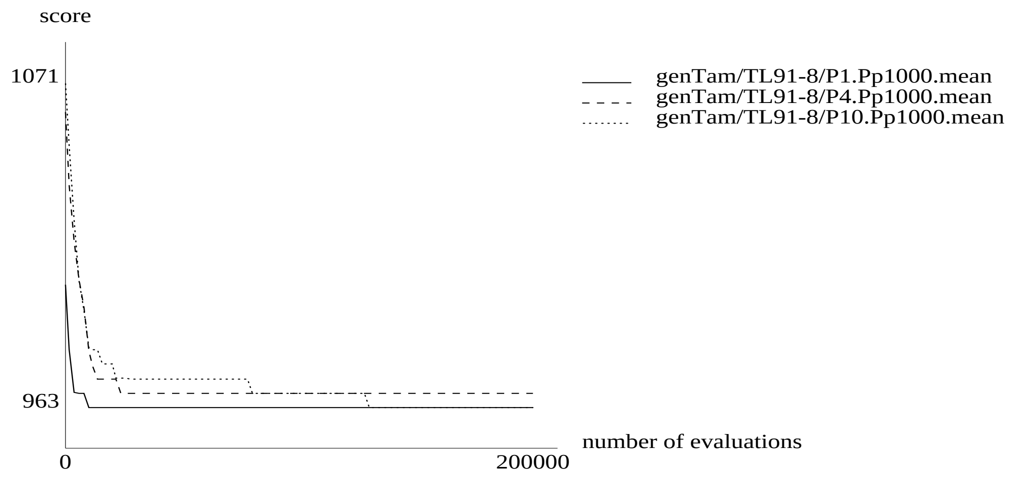


Figure E.7. A comparison of each population number in TL91-7





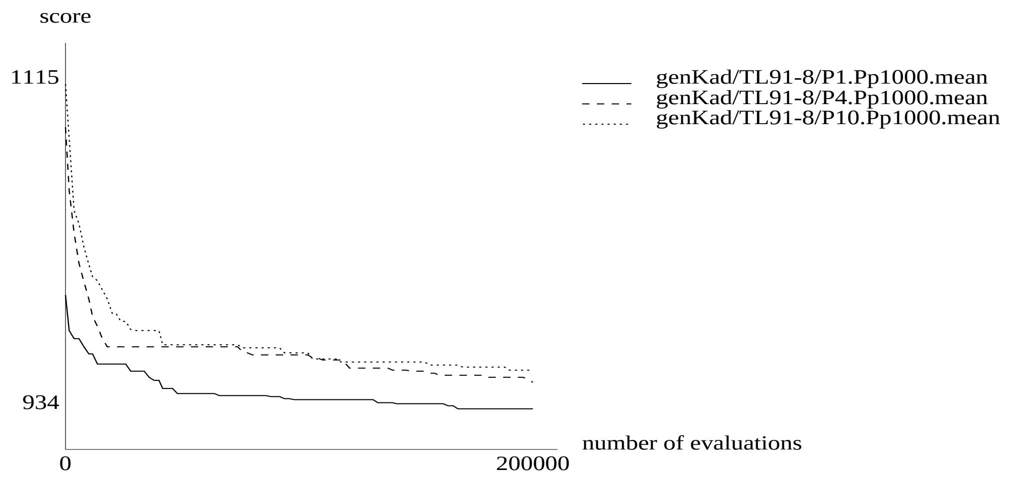


Figure E.8. A comparison of each population number in TL91-8

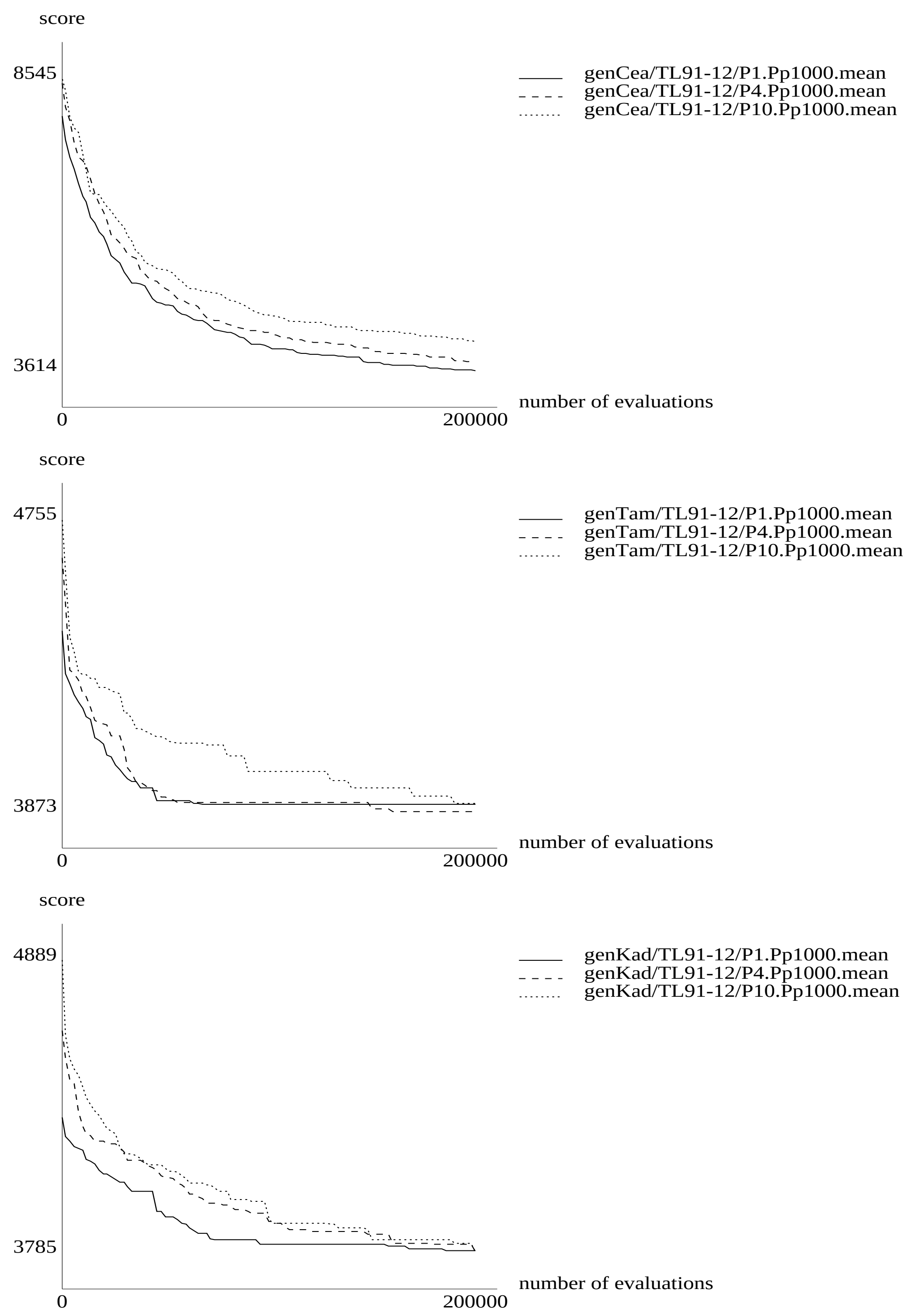
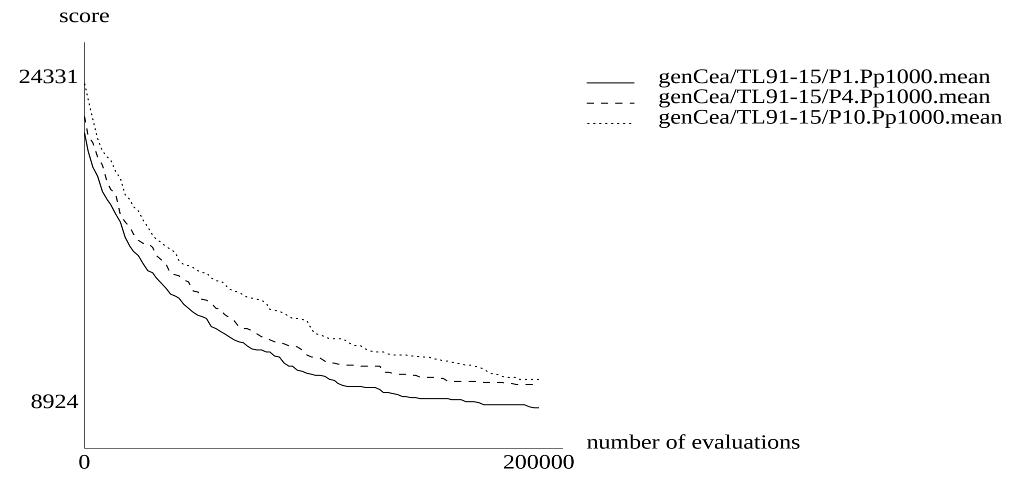
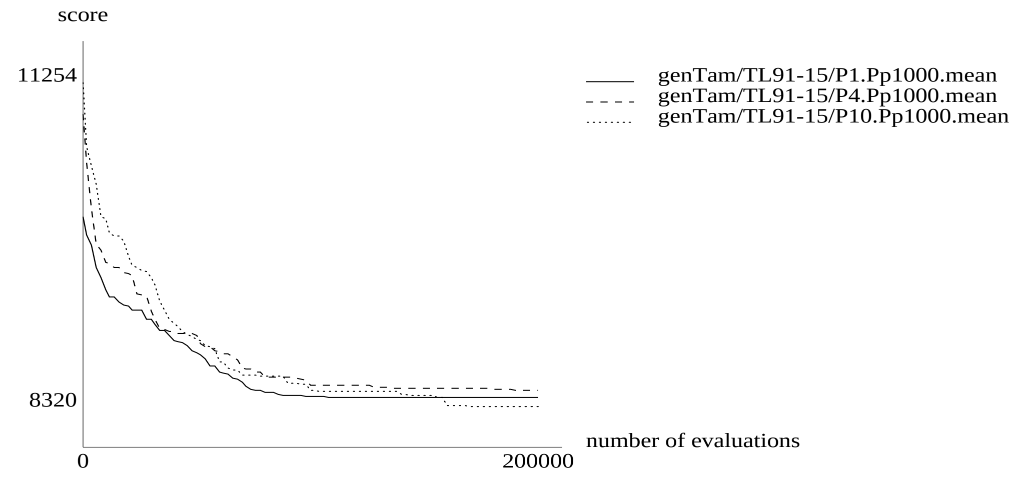


Figure E.9. A comparison of each population number in TL91-12





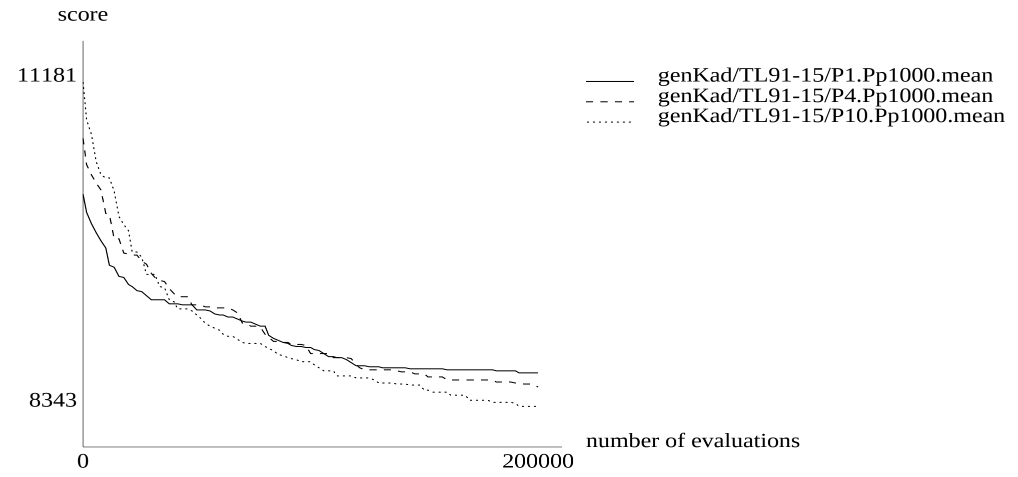
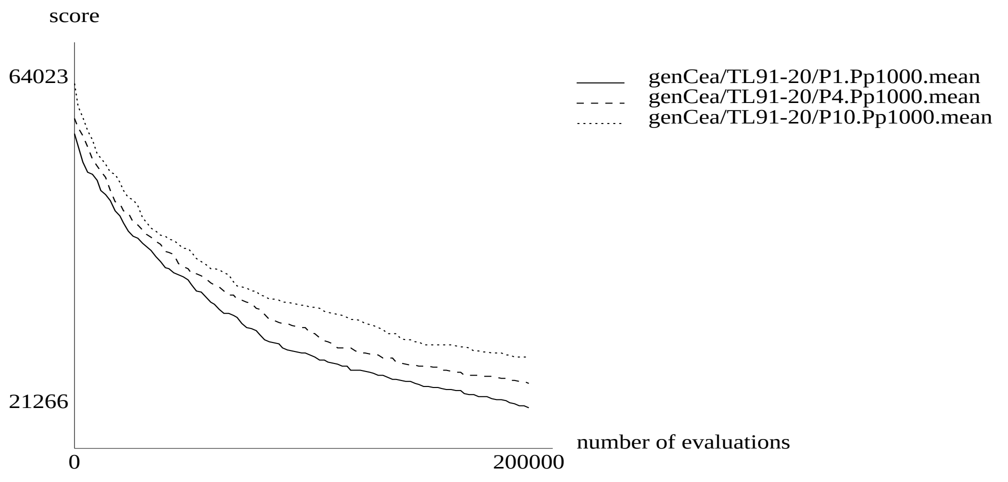
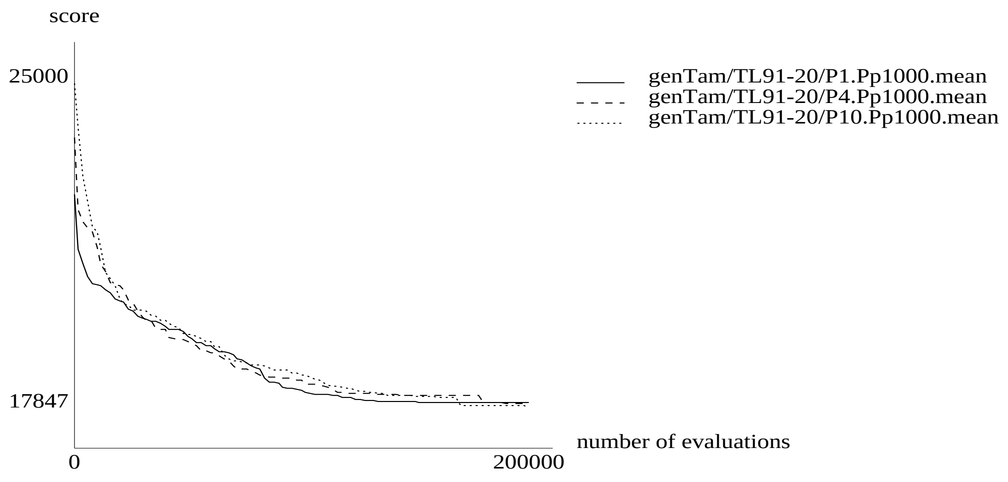


Figure E.10. A comparison of each population number in TL91-15





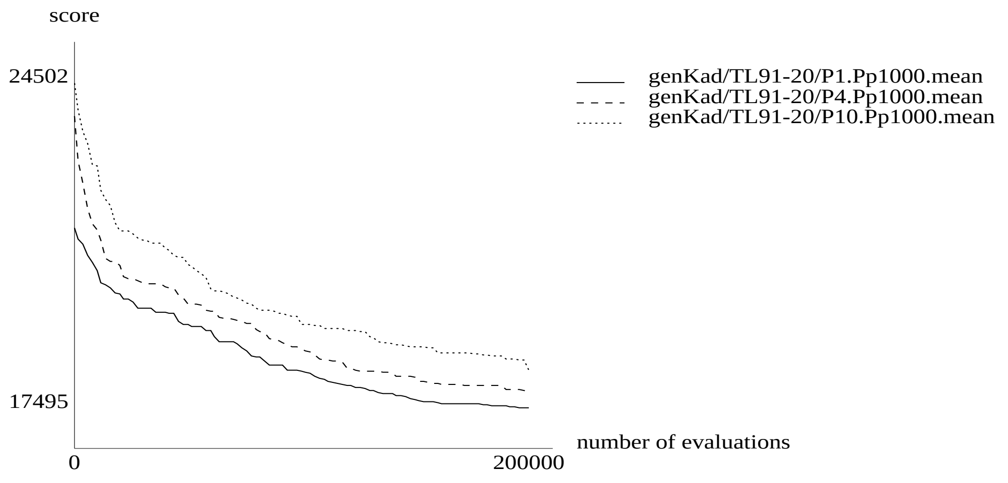


Figure E.11. A comparison of each population number in TL91-20

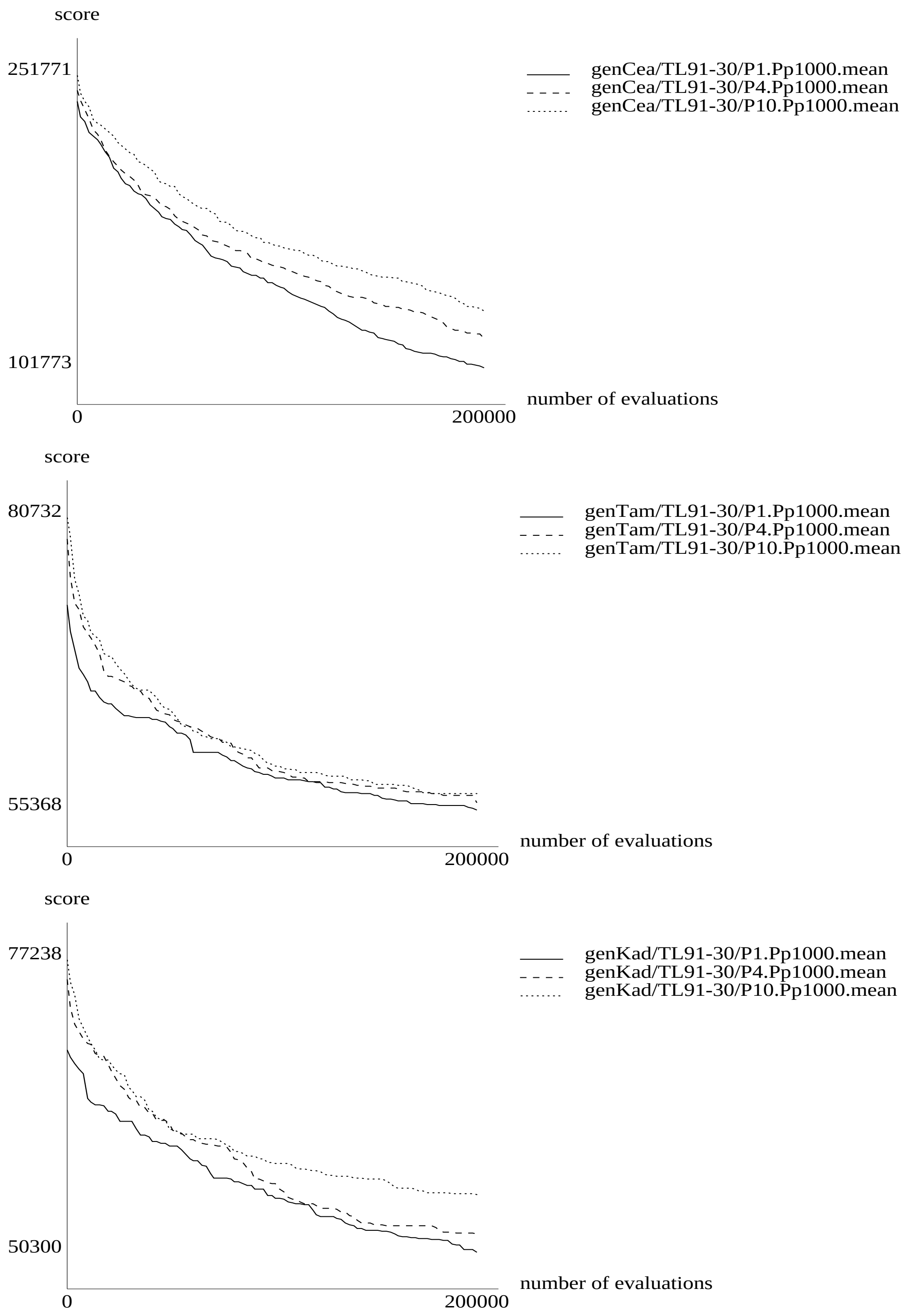
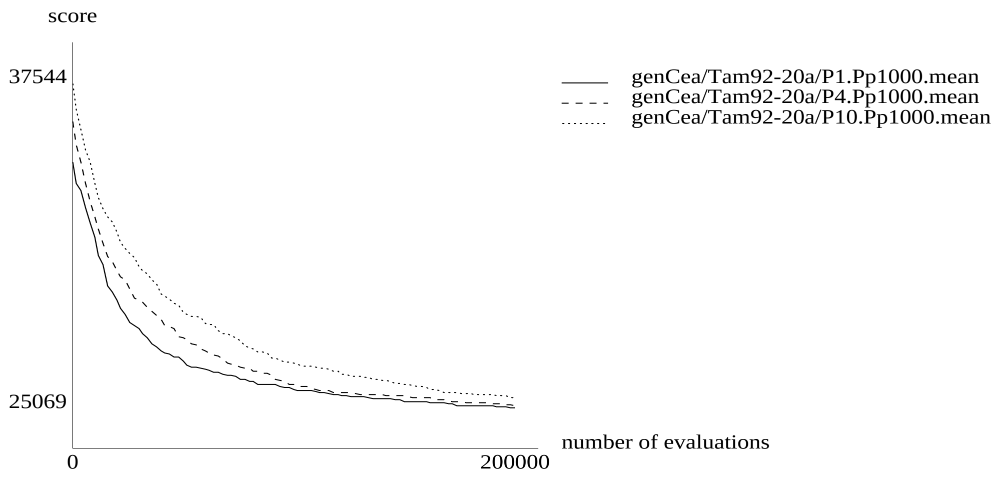
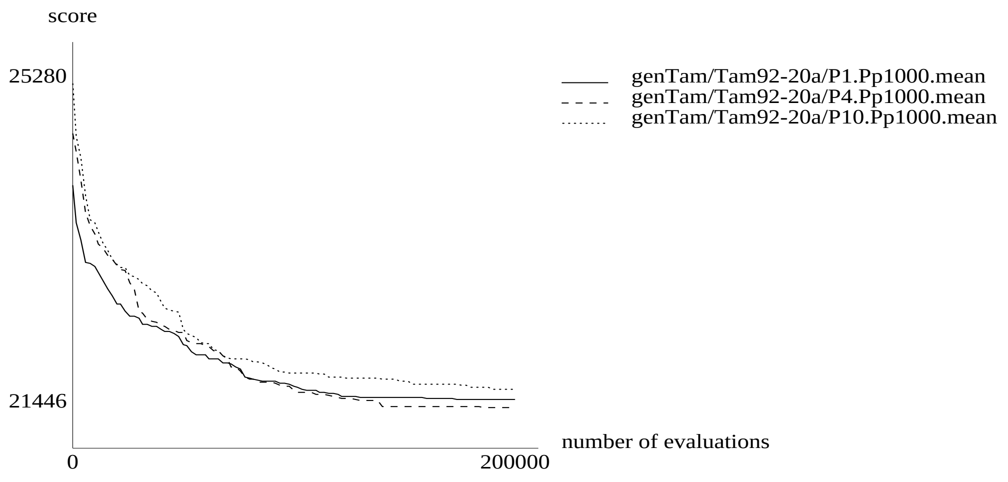


Figure E.12. A comparison of each population number in TL91-30





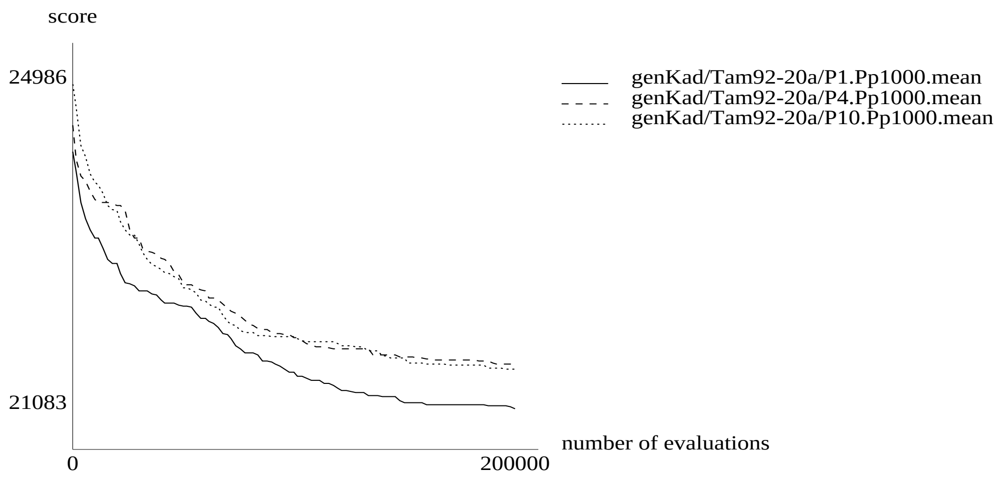
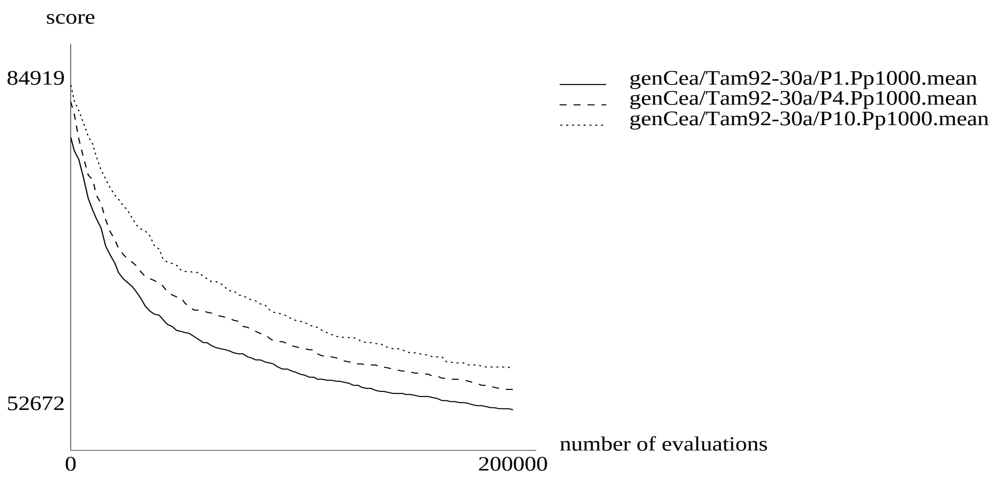
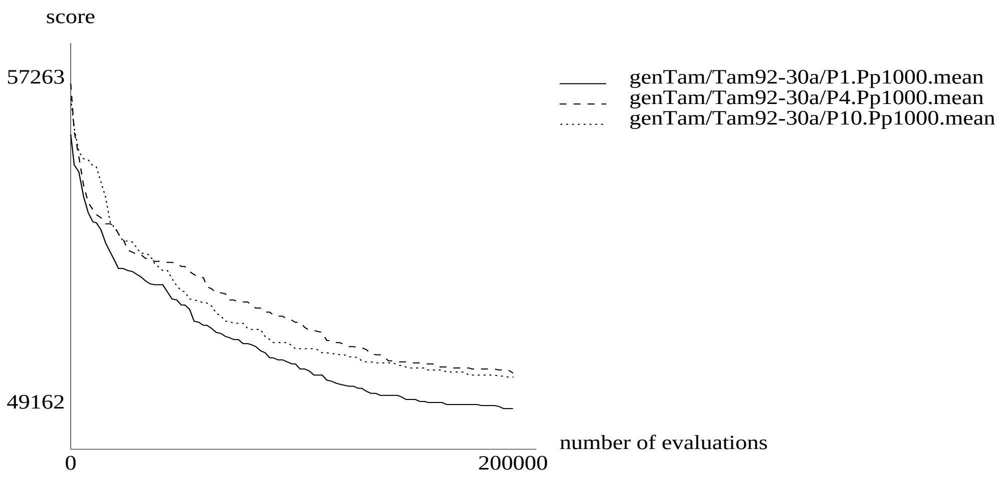


Figure E.13. A comparison of each population number in Tam92-20a





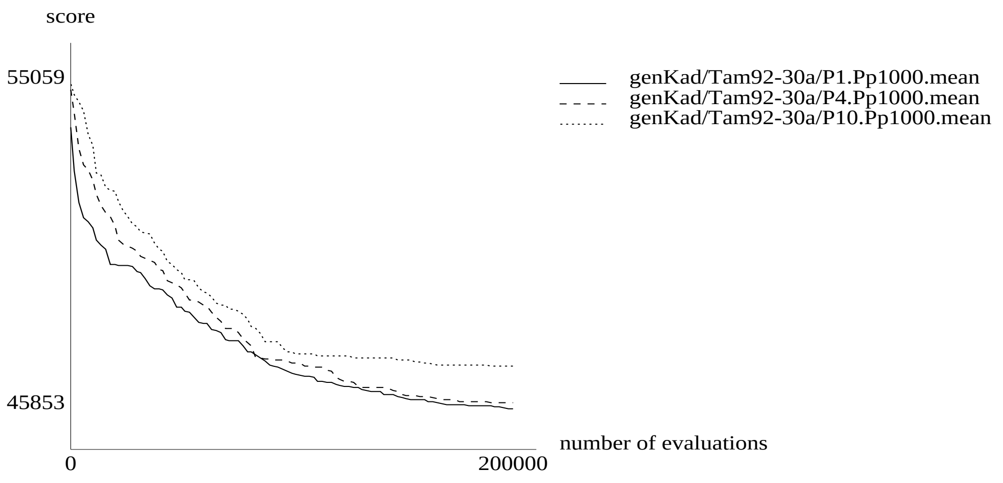
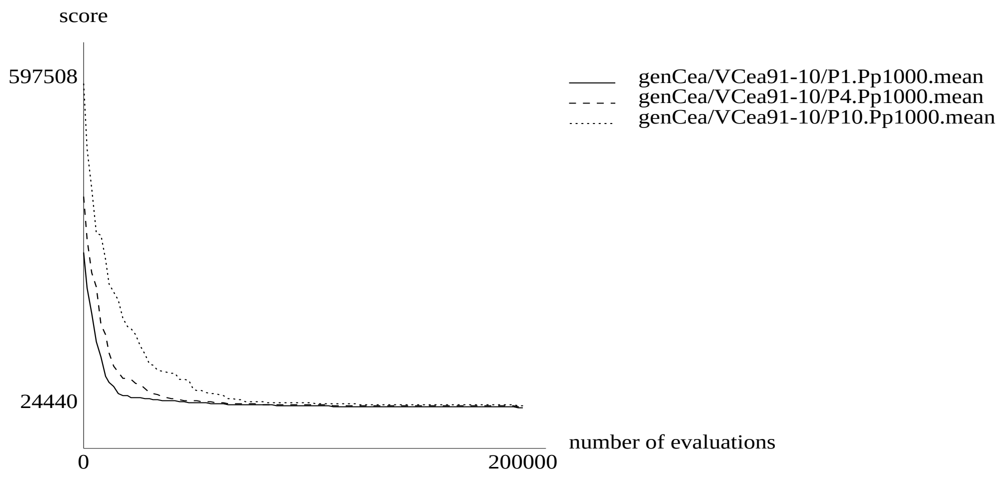
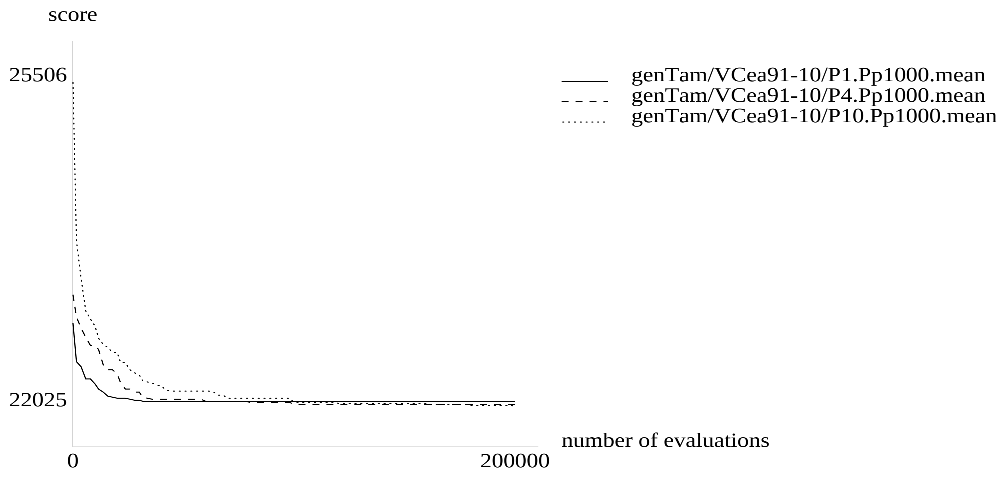


Figure E.14. A comparison of each population number in Tam92-30a





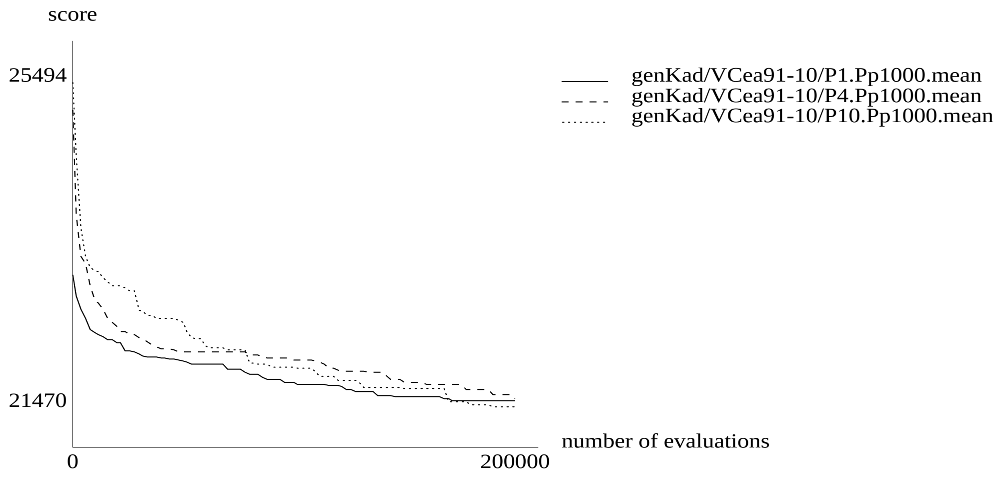


Figure E.15. A comparison of each population number in VCea91-10

Appendix F

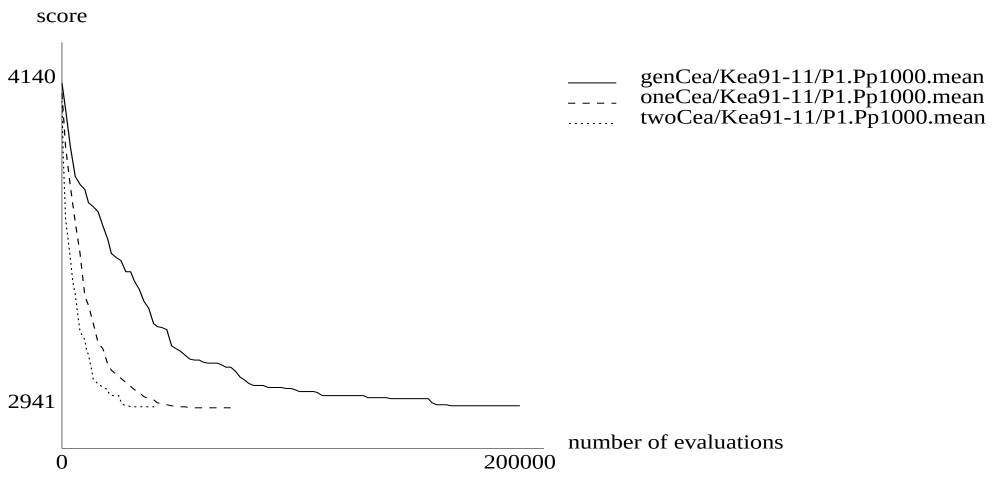
The Results of Reproduction Investigation

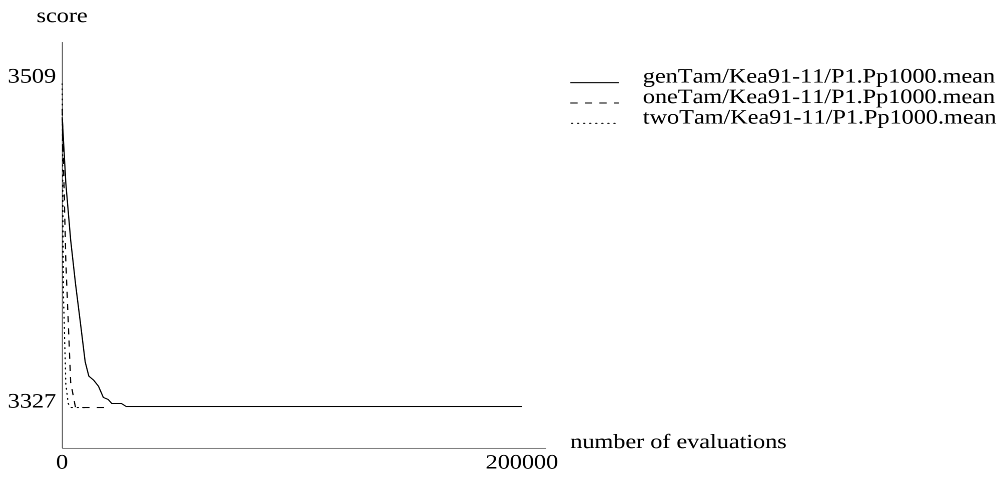
This appendix shows the state of convergence of GAs to compare the reproduction methods. The name of each GA is denoted as follows.

$rrraaa/fff/\texttt{P}ggg\texttt{.P}ppp$

where rrr	$=$	reproduction method (gen, one, two)
aaa	$=$	algorithm (Cea, Tam, DK, Tam2, DK2, Kad)
fff	$=$	the name of FLP (Kea91-11, TL91-5, etc.)
ggg	$=$	the number of populations (1, 4, 10)
ppp	$=$	population size $\times$ the number of populations (50, 200, 1000)

The horizontal axis indicates the number of evaluations, whereas the vertical axis indicates the mean of the best individual scores. Here, smaller score is better.





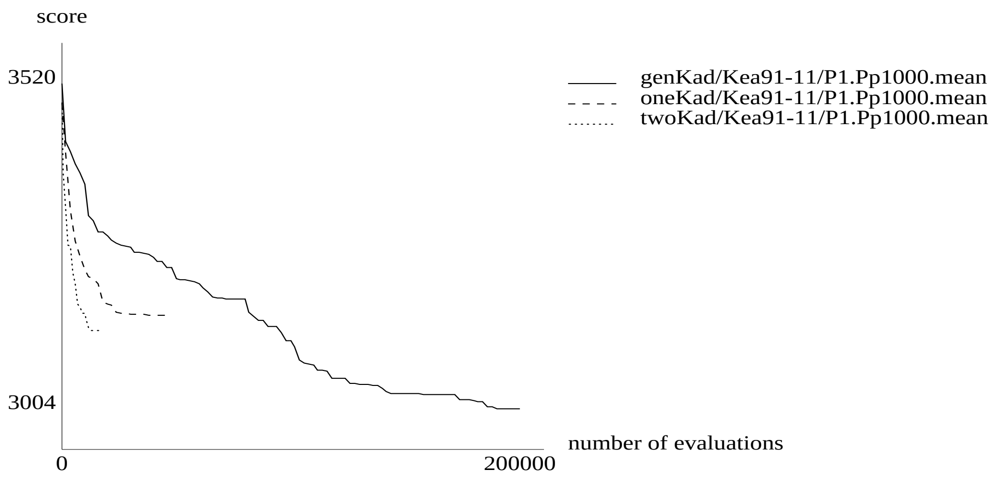


Figure F.1. A comparison of each reproduction method in Kea91-11

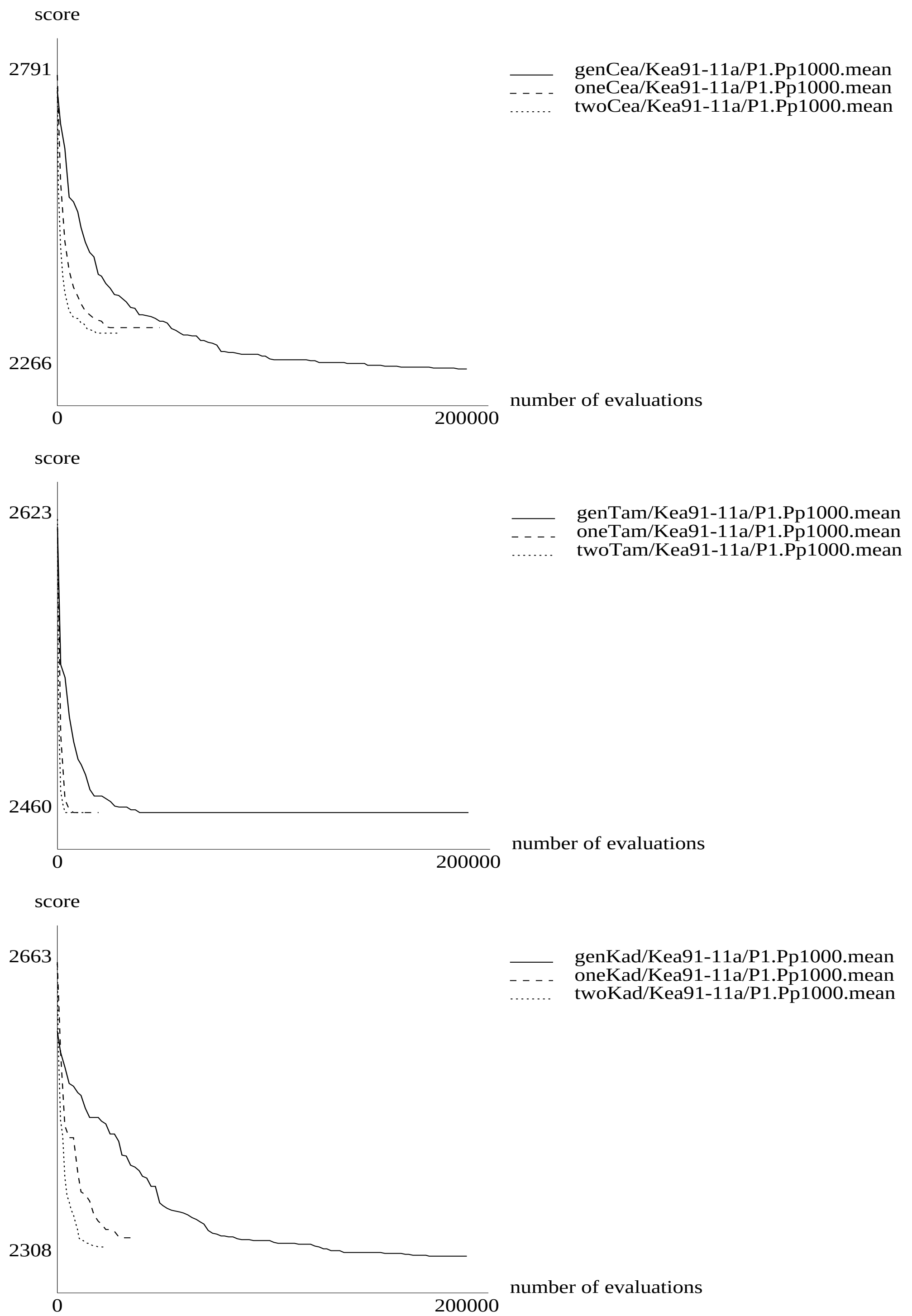
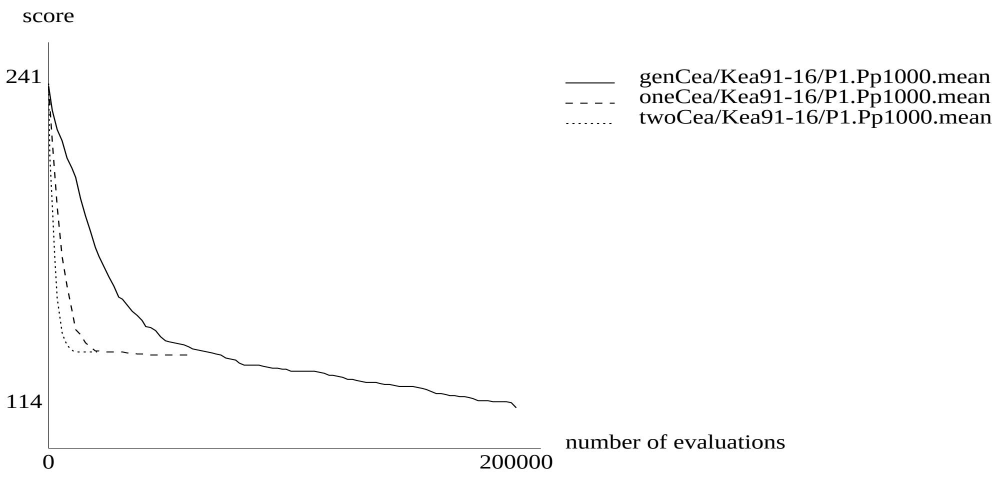
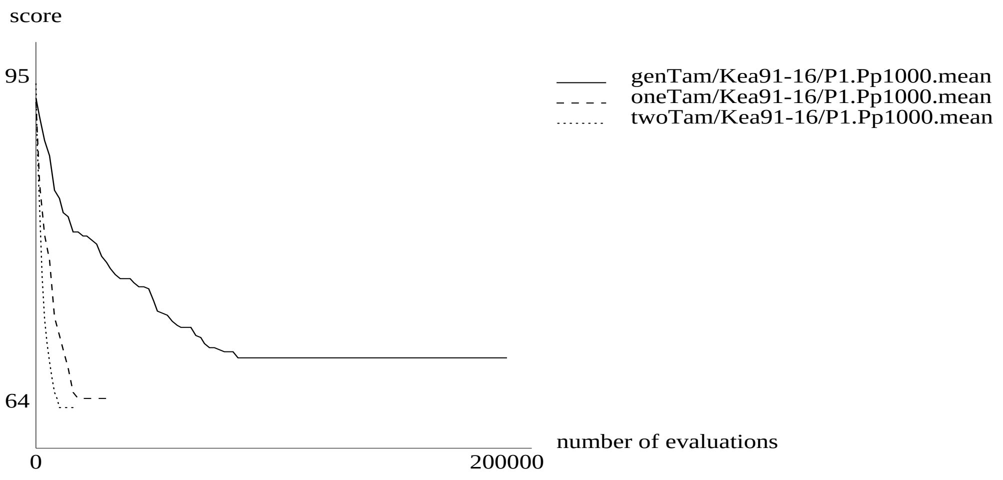


Figure F.2. A comparison of each reproduction method in Kea91-11a





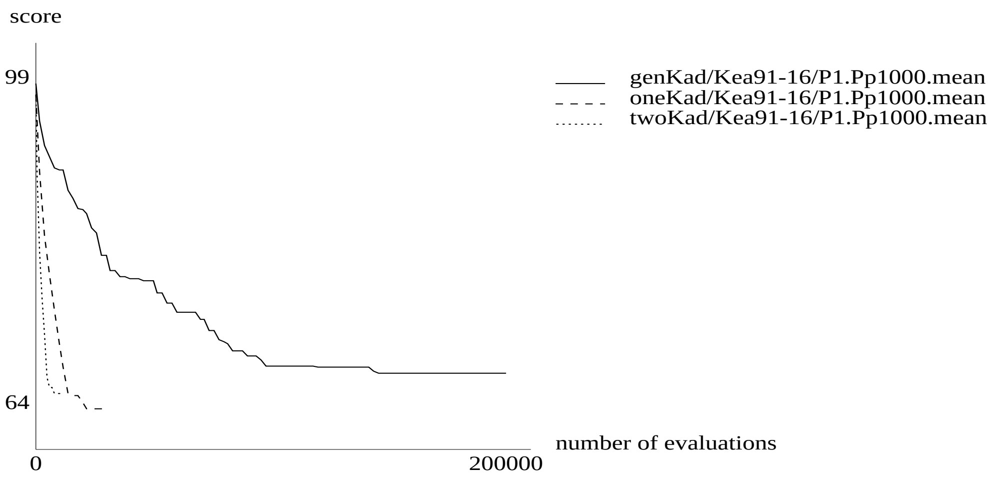


Figure F.3. A comparison of each reproduction method in Kea91-16

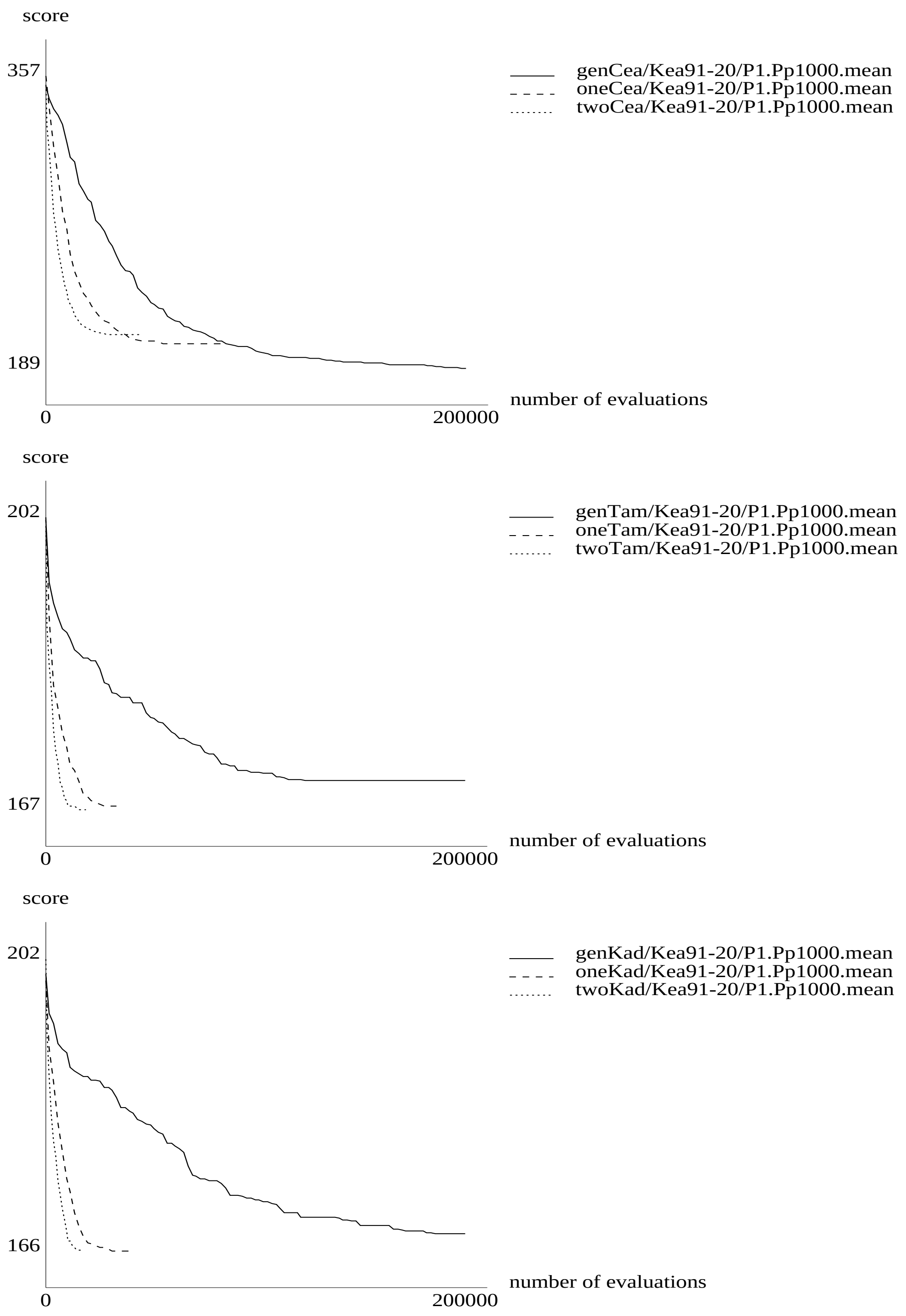
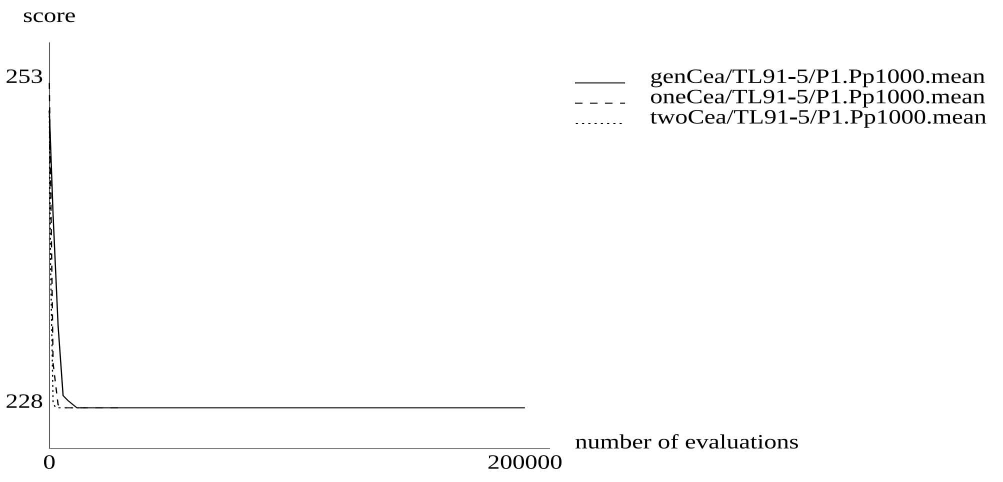
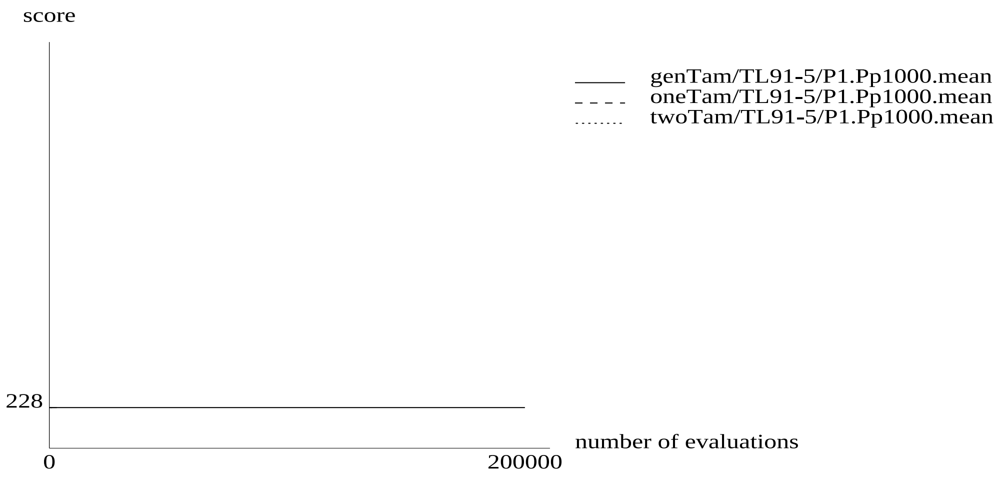


Figure F.4. A comparison of each reproduction method in Kea91-20





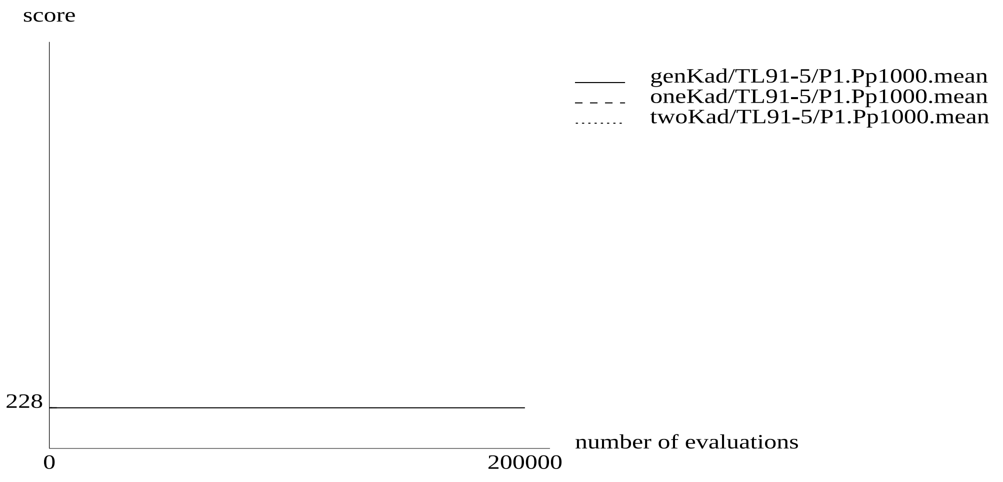
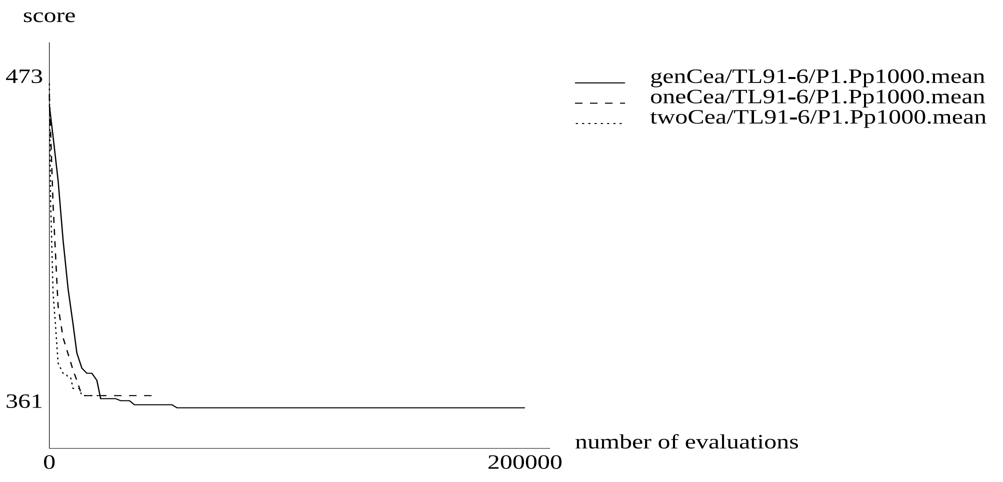
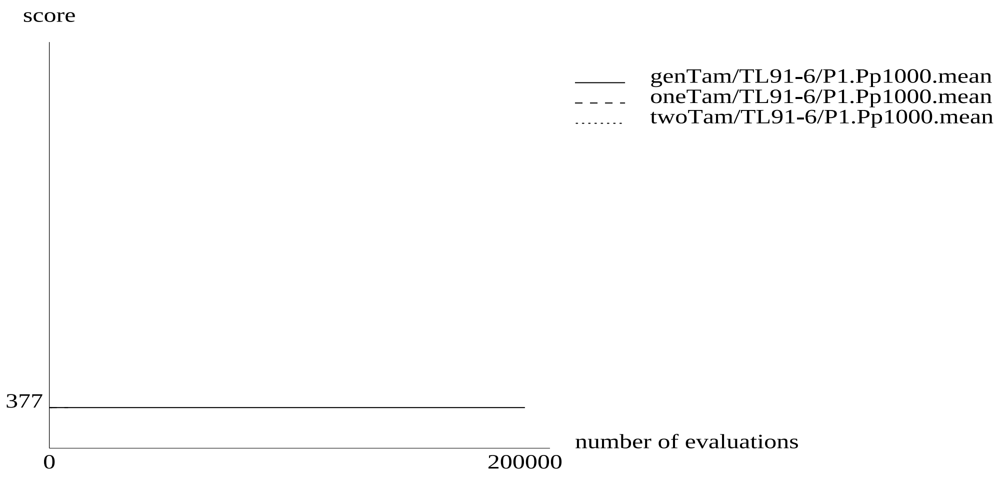


Figure F.5. A comparison of each reproduction method in TL91-5





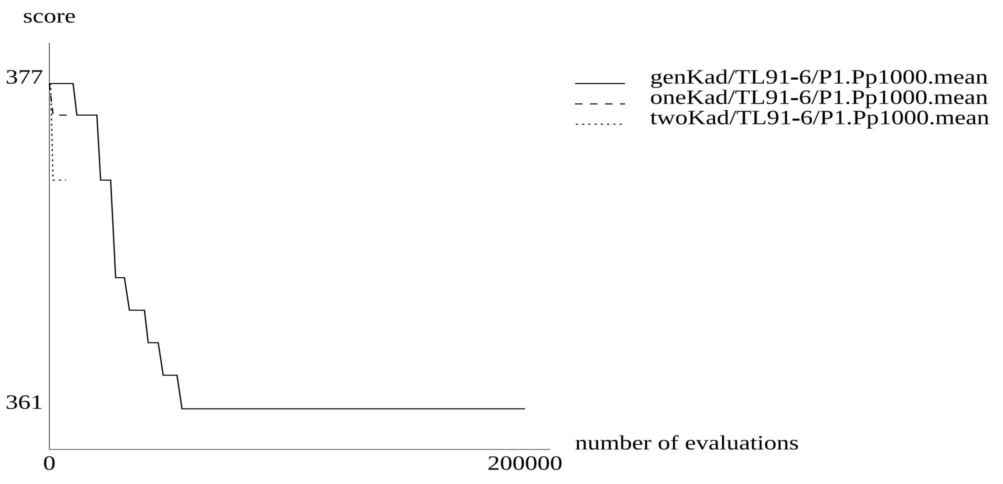
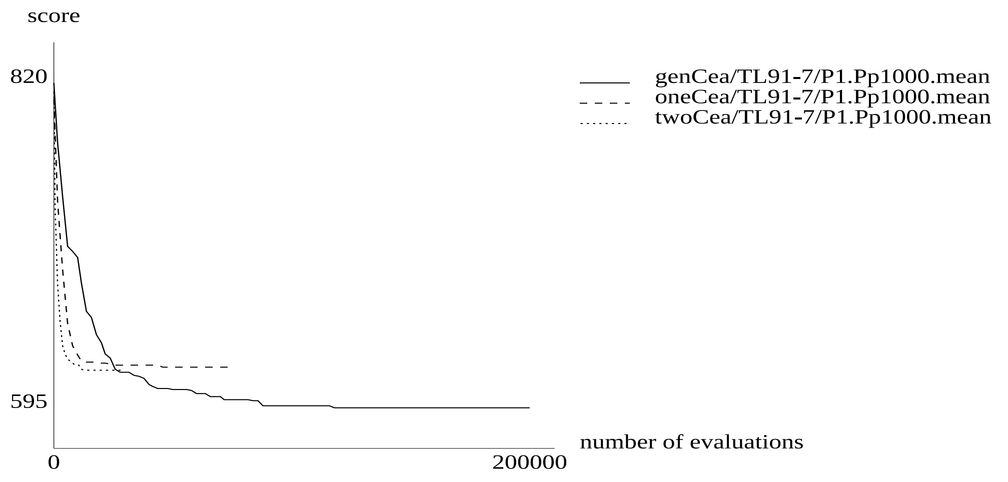
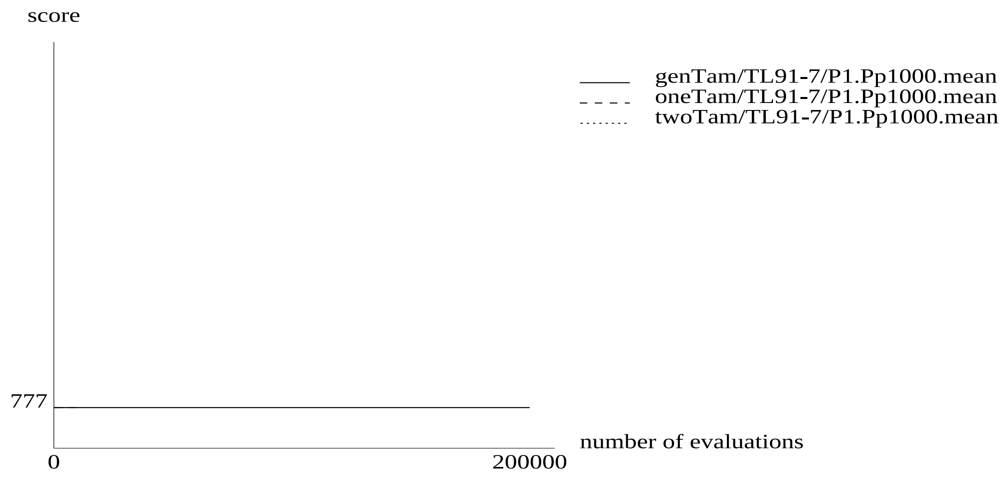


Figure F.6. A comparison of each reproduction method in TL91-6





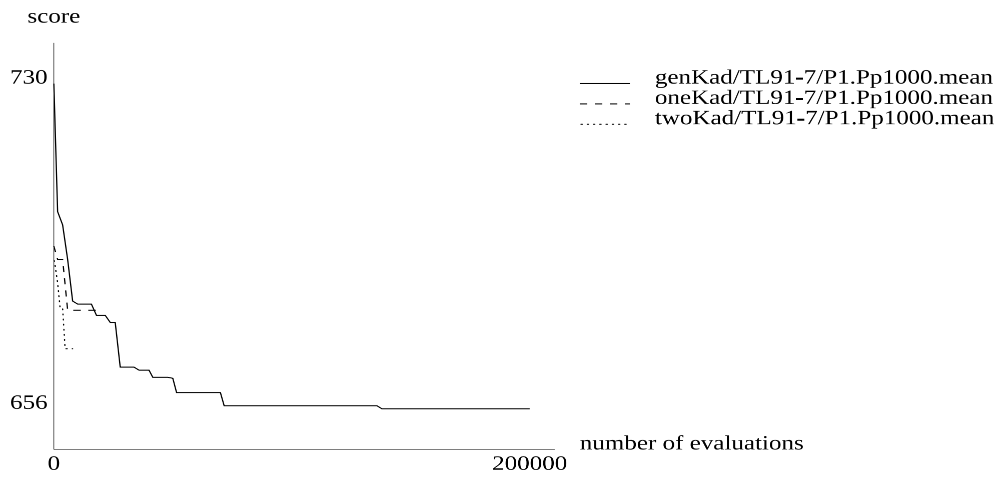
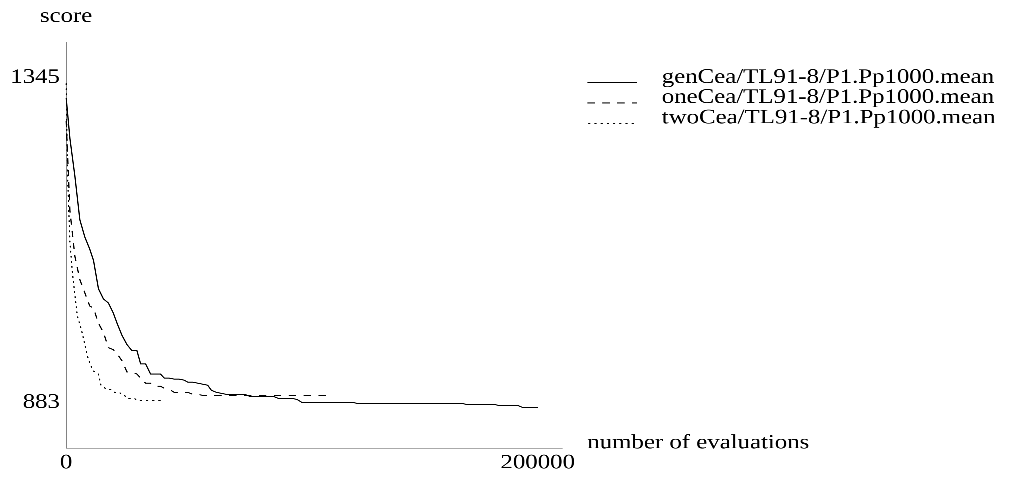
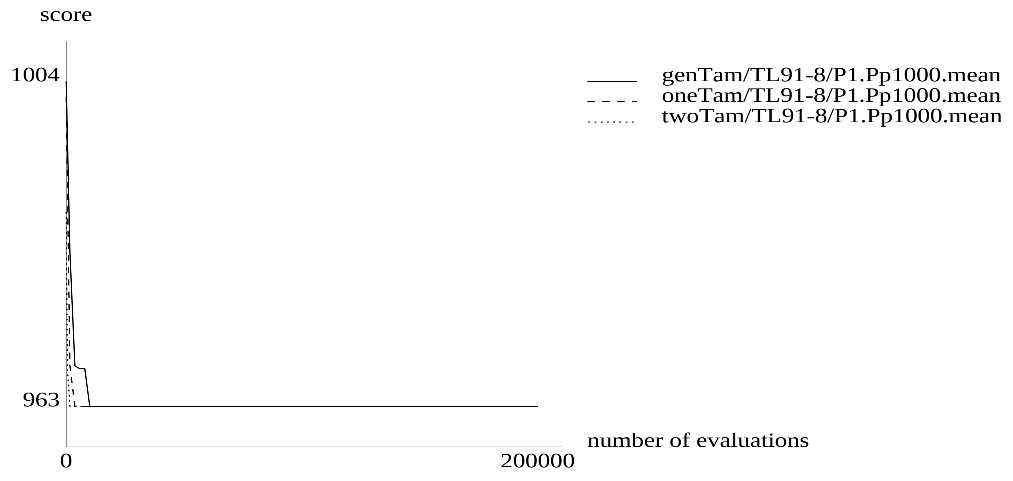


Figure F.7. A comparison of each reproduction method in TL91-7





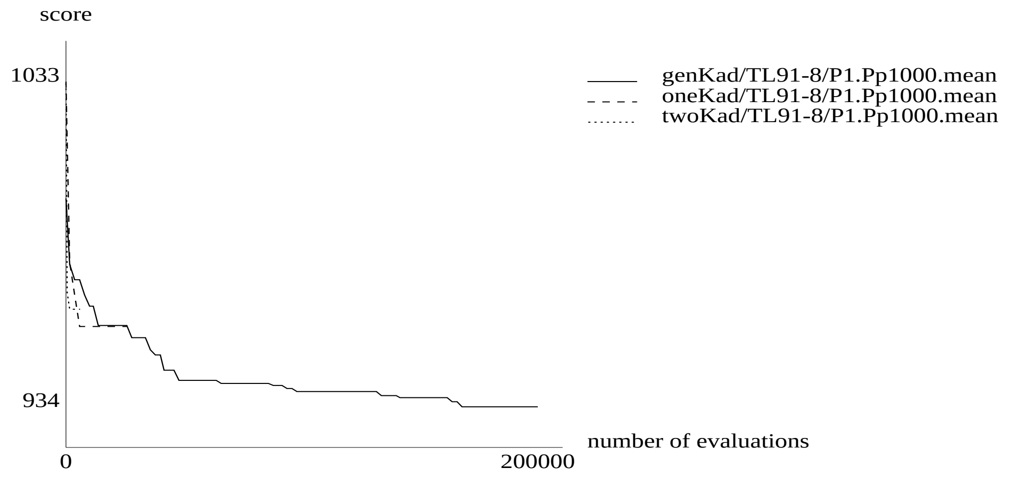
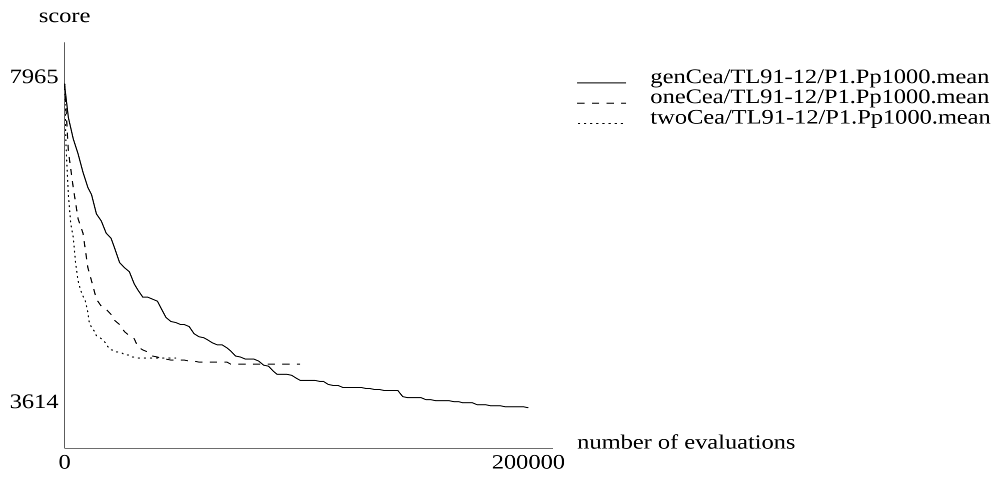
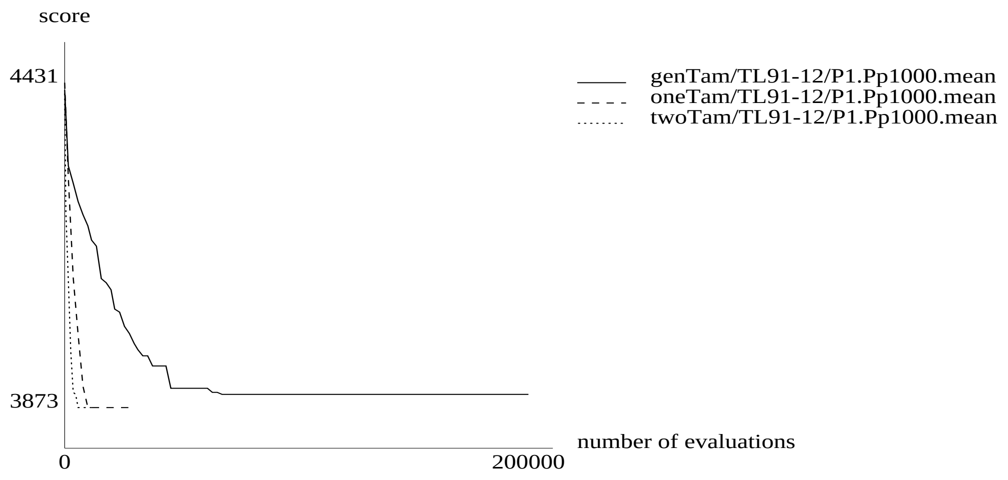


Figure F.8. A comparison of each reproduction method in TL91-8





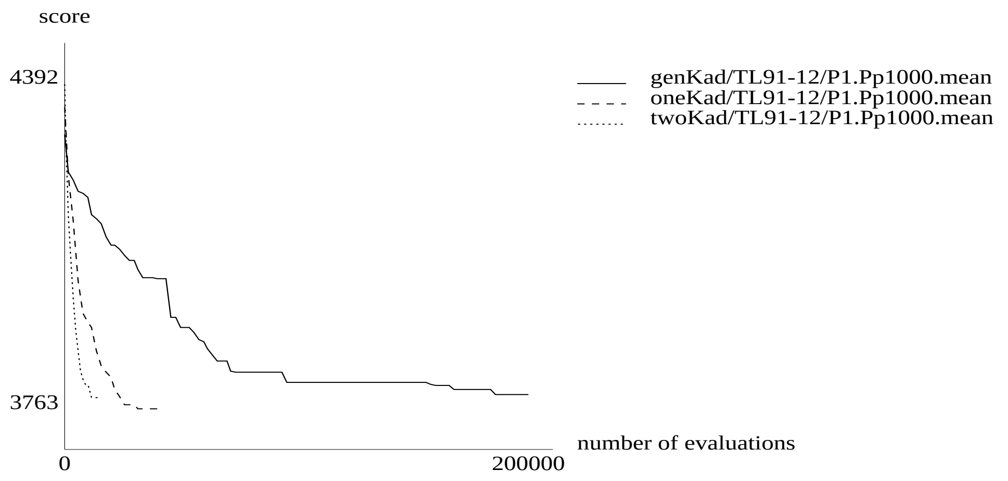


Figure F.9. A comparison of each reproduction method in TL91-12

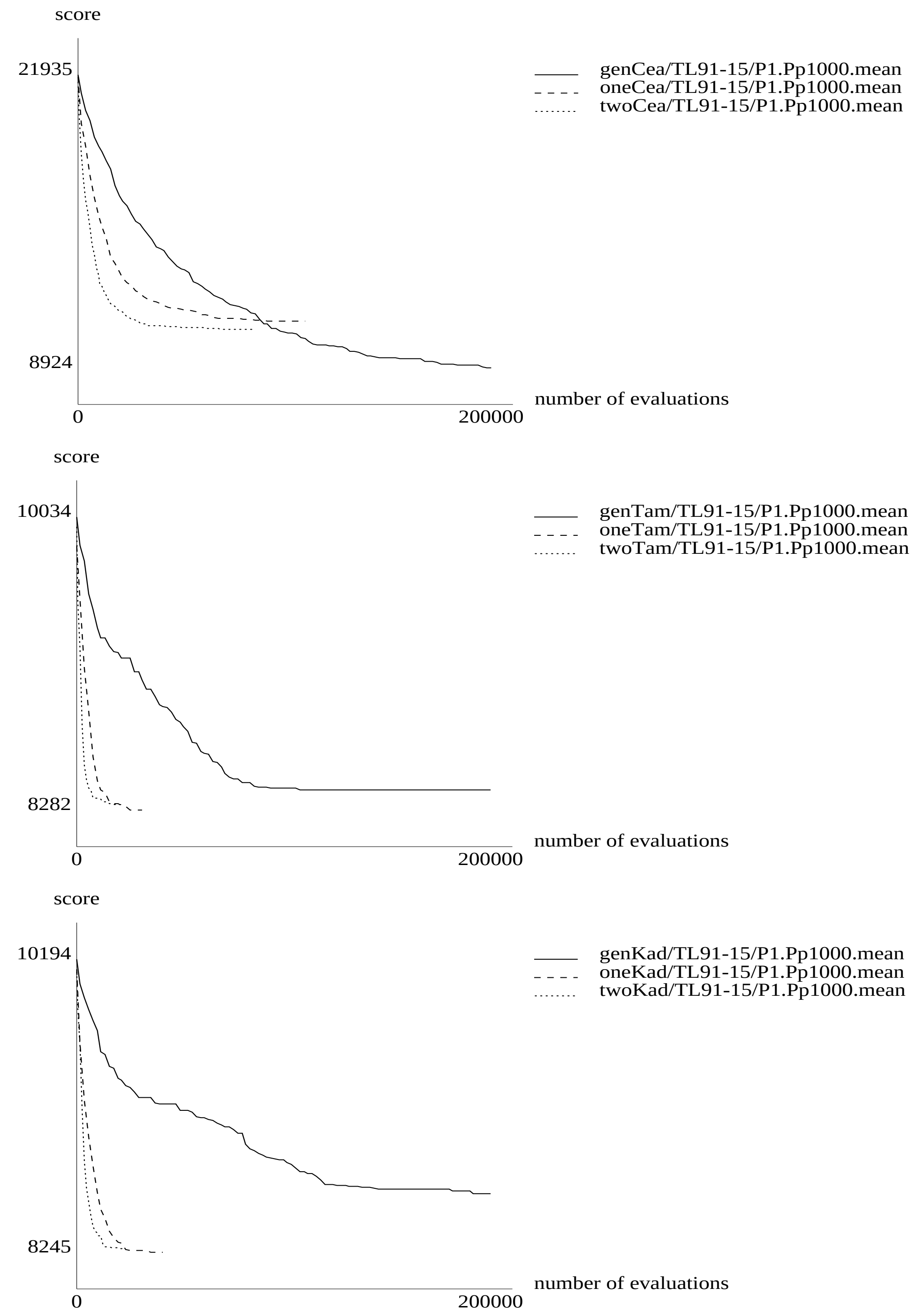
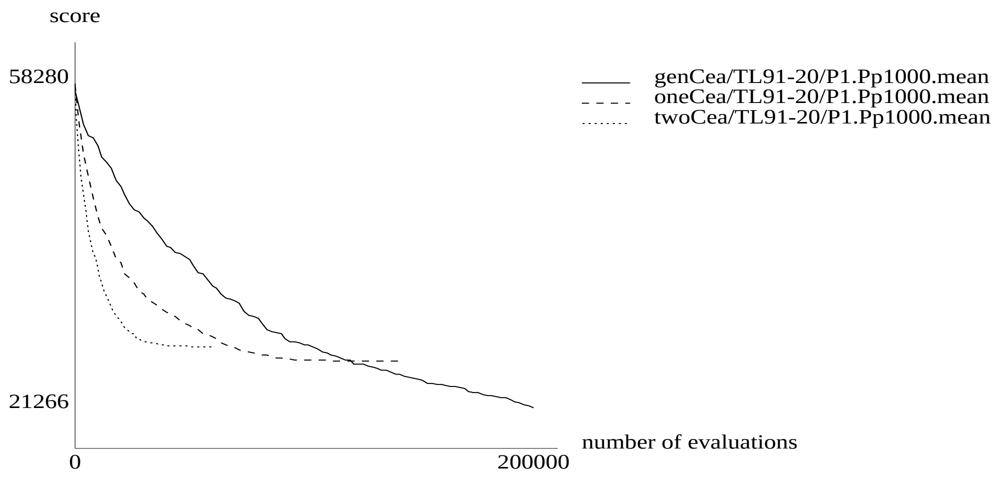
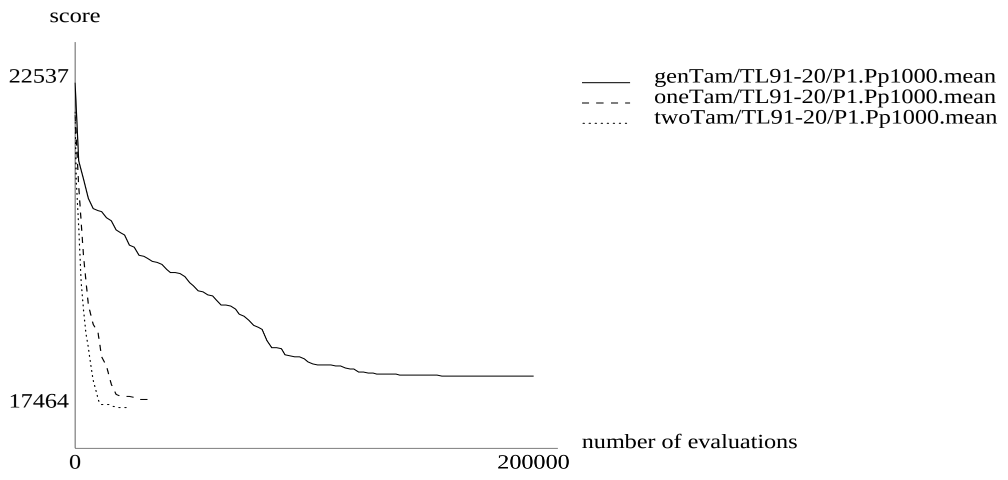


Figure F.10. A comparison of each reproduction method in TL91-15





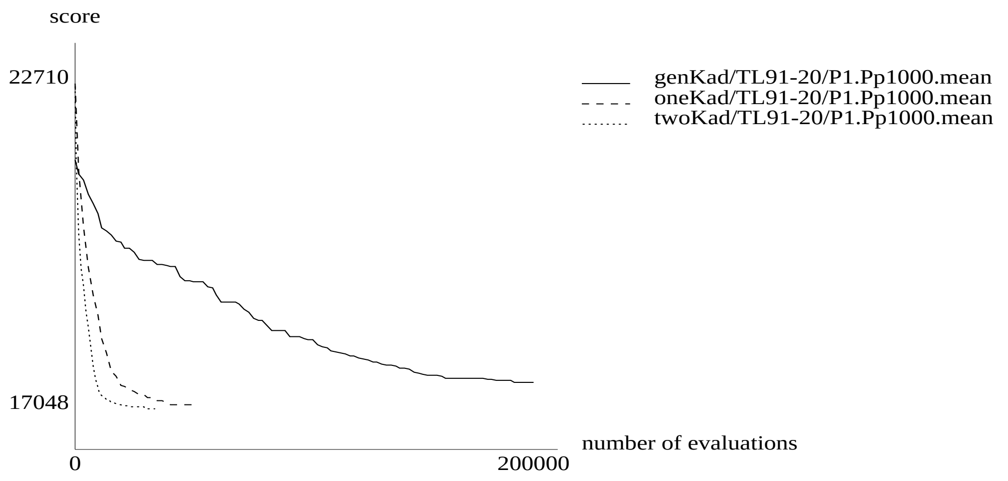
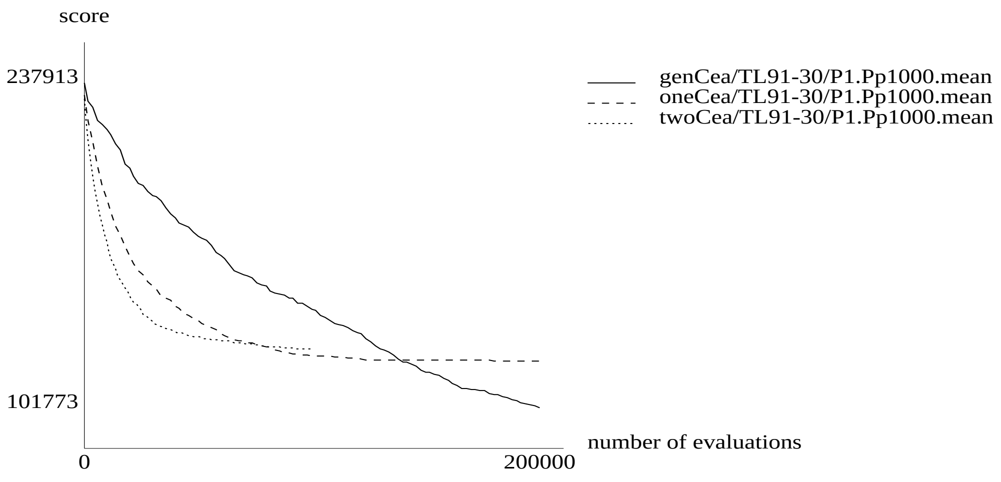
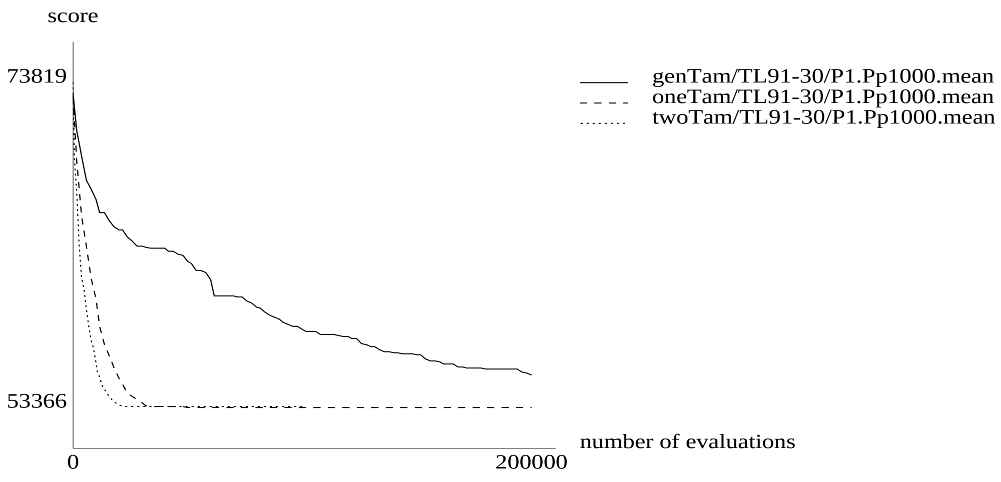


Figure F.11. A comparison of each reproduction method in TL91-20





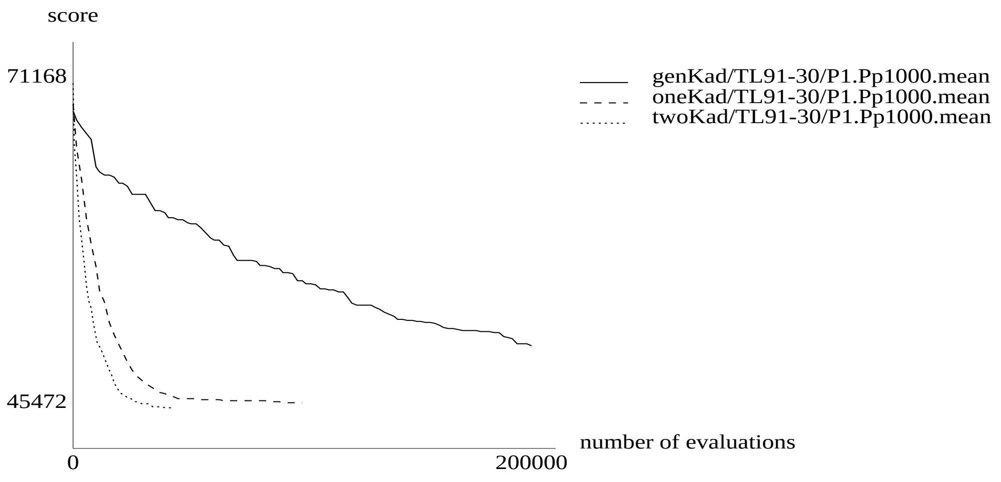


Figure F.12. A comparison of each reproduction method in TL91-30

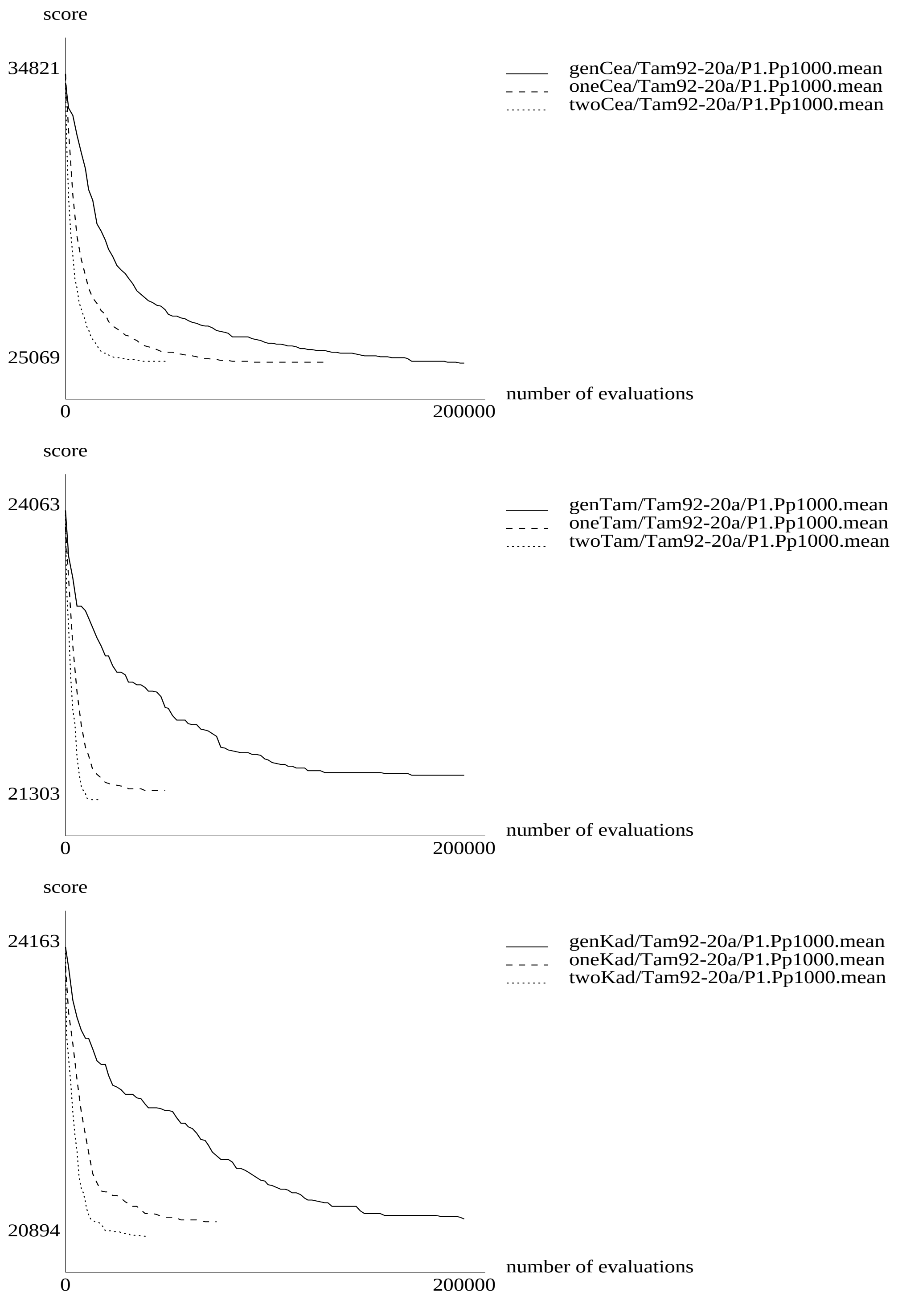
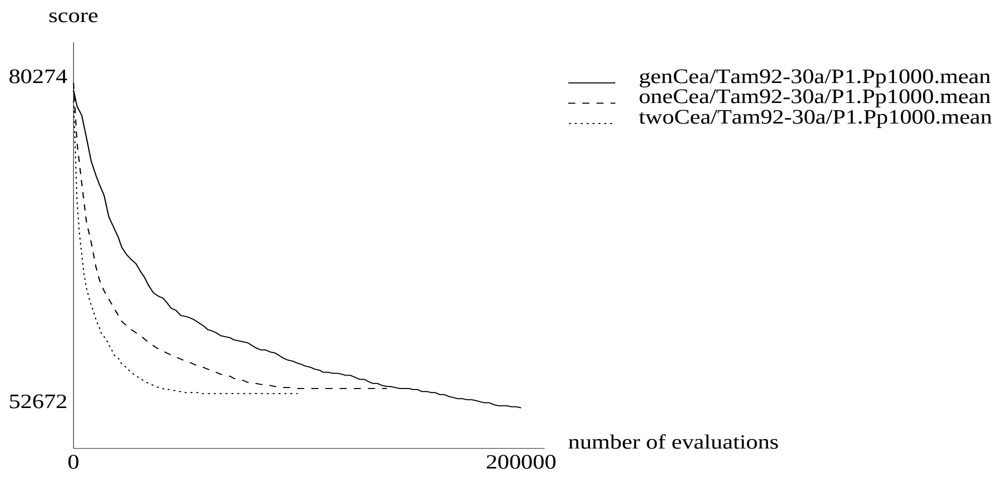
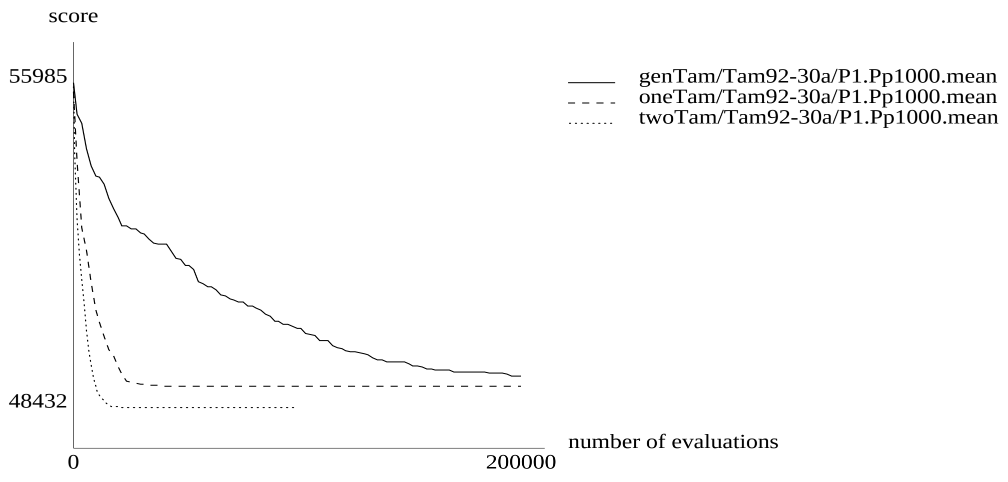


Figure F.13. A comparison of each reproduction method in Tam92-20a





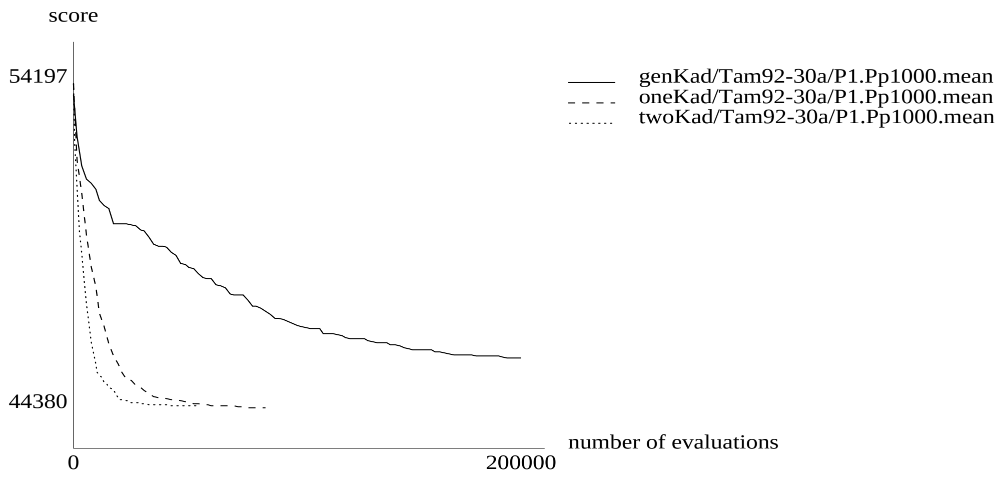
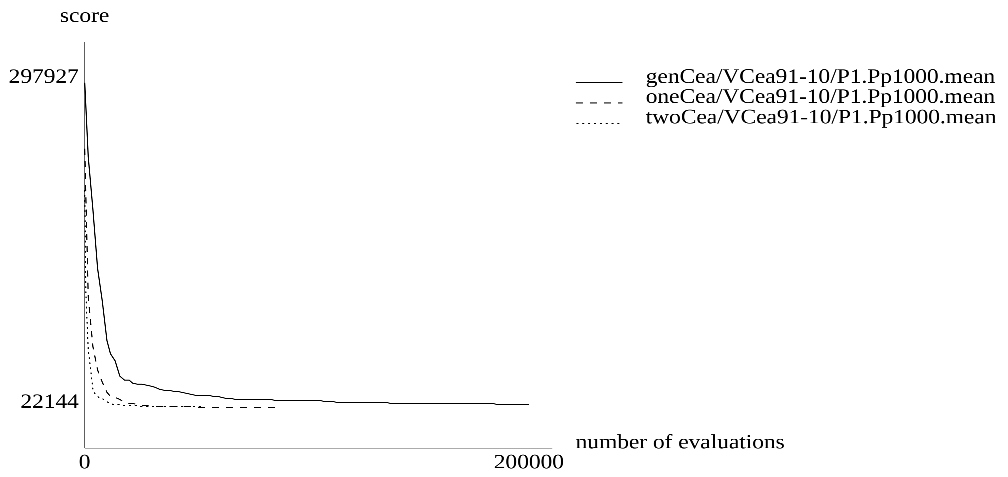
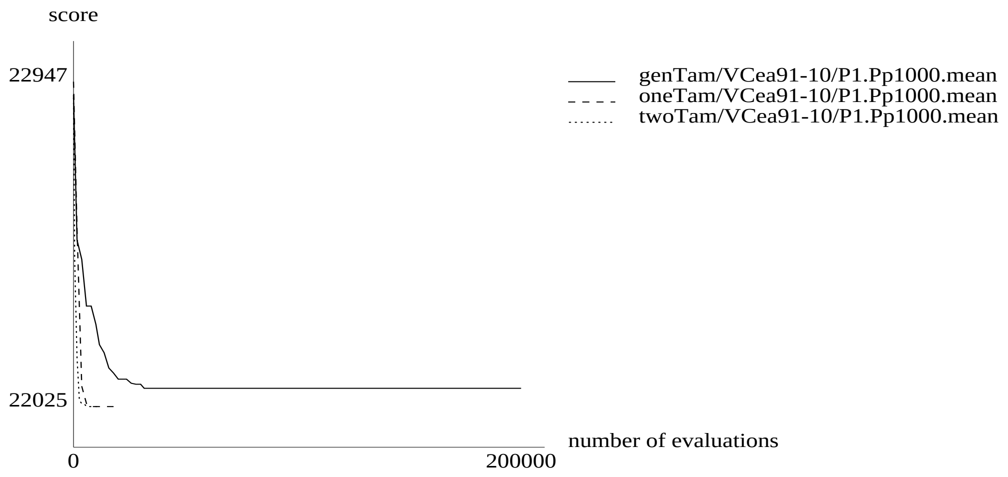


Figure F.14. A comparison of each reproduction method in Tam92-30a





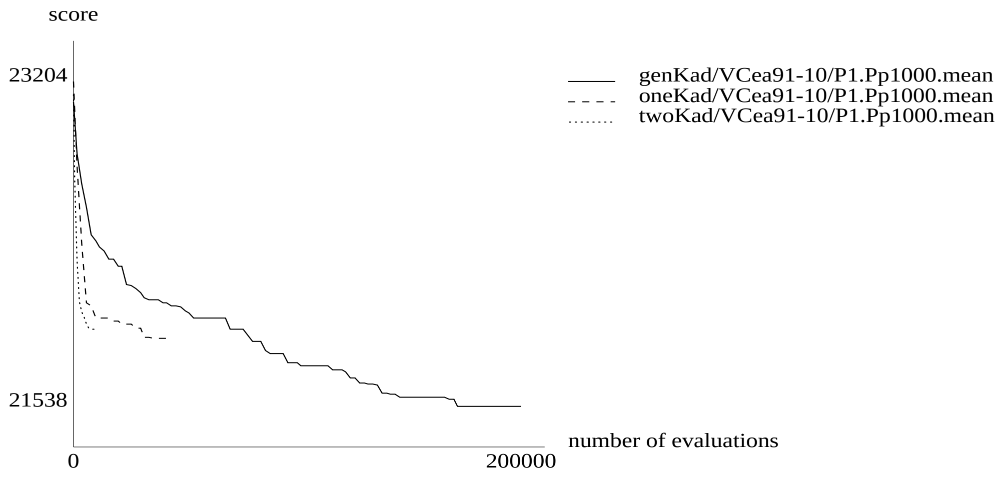


Figure F.15. A comparison of each reproduction method in VCea91-10

Appendix G

The Best Layouts

Figures G.1 to G.16 show the best physical layout of each FLP generated by GAs. And, Figure G.17 summarises the GA's names producing the layouts, the scores of the layouts, and the chromosomes representing the layouts.

As for Tam92-20a, the best layout includes a fatal part where a facility (No.6) is assigned to two regions separated by a prespecified area. So, I attached the second best layout obtained by another GA, for reference. Incidentally, this layout still gets better score than the best layouts reported in previous papers.

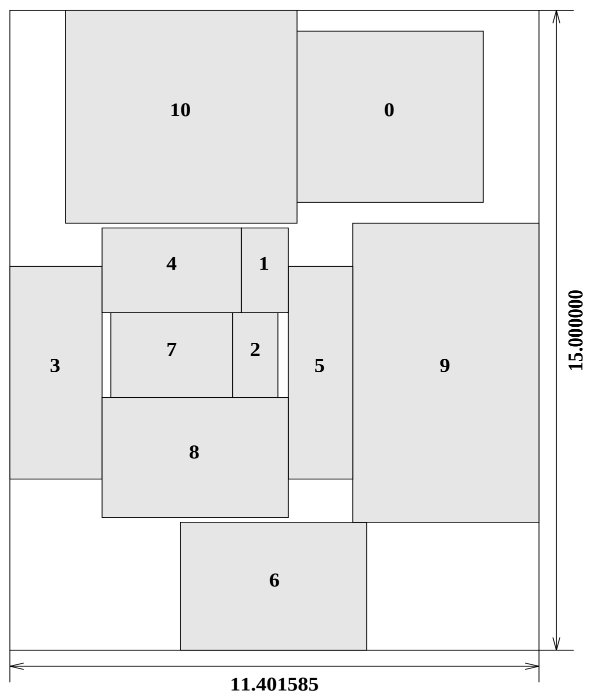


Figure G.1. The best layout for Kea91-11

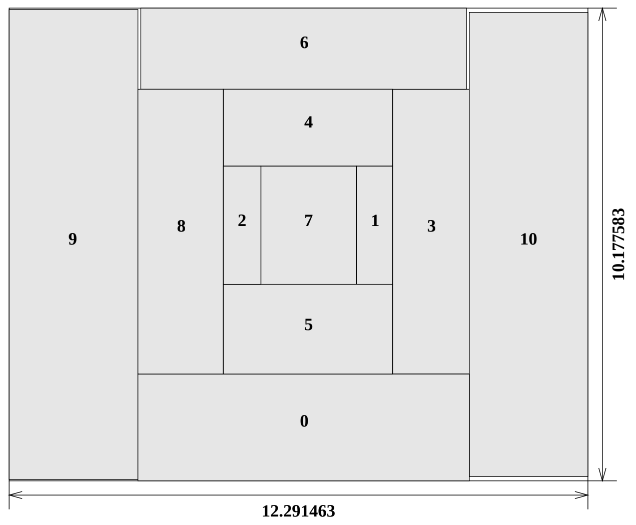


Figure G.2. The best layout for Kea91-11a

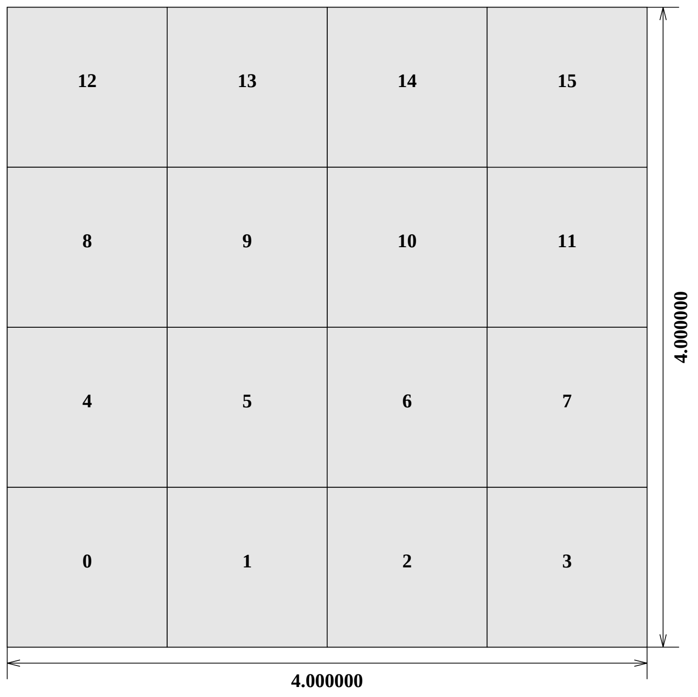


Figure G.3. The best layout for Kea91-16

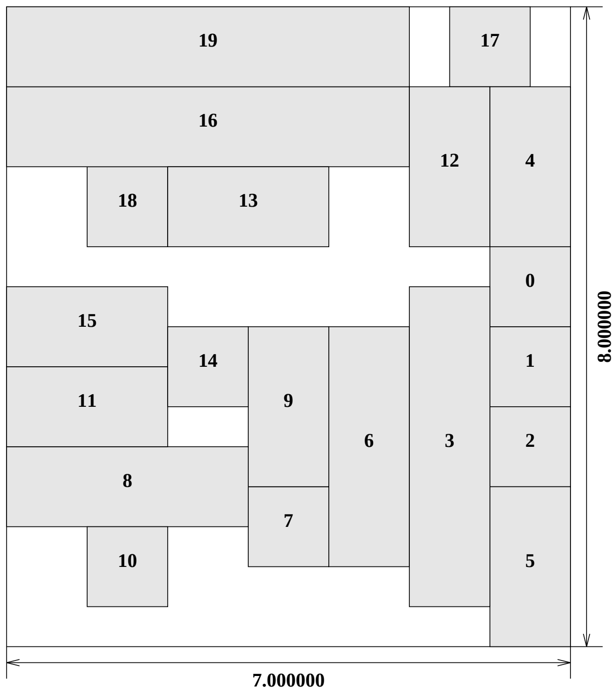


Figure G.4. The best layout for Kea91-20

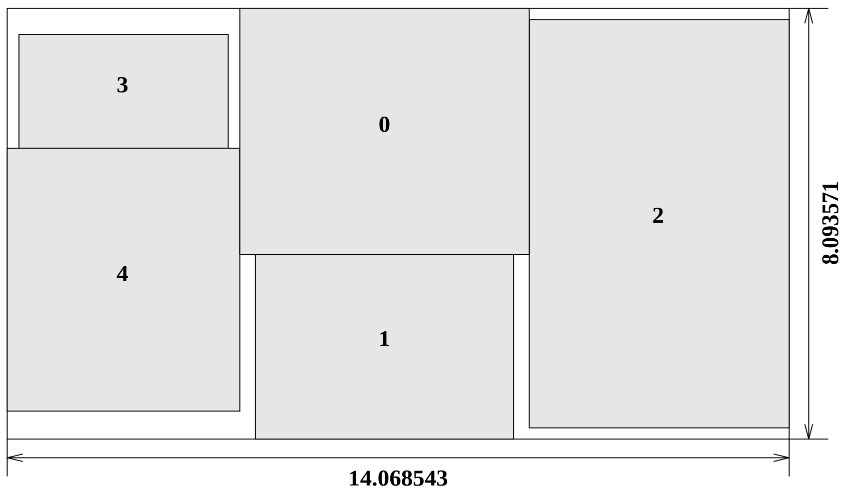


Figure G.5. The best layout for TL91-5

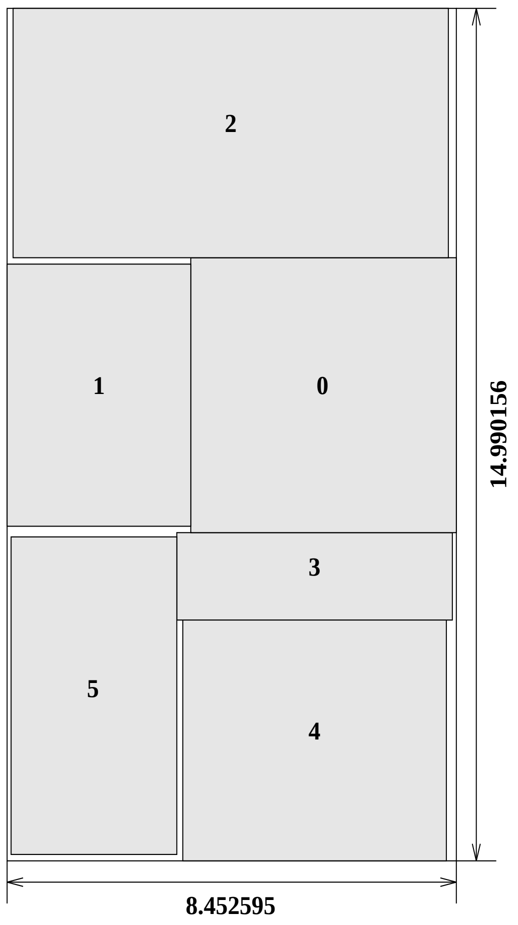


Figure G.6. The best layout for TL91-6

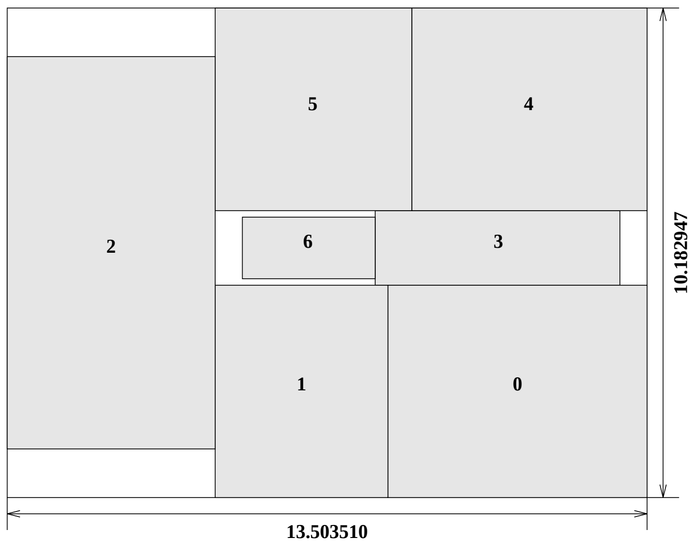


Figure G.7. The best layout for TL91-7

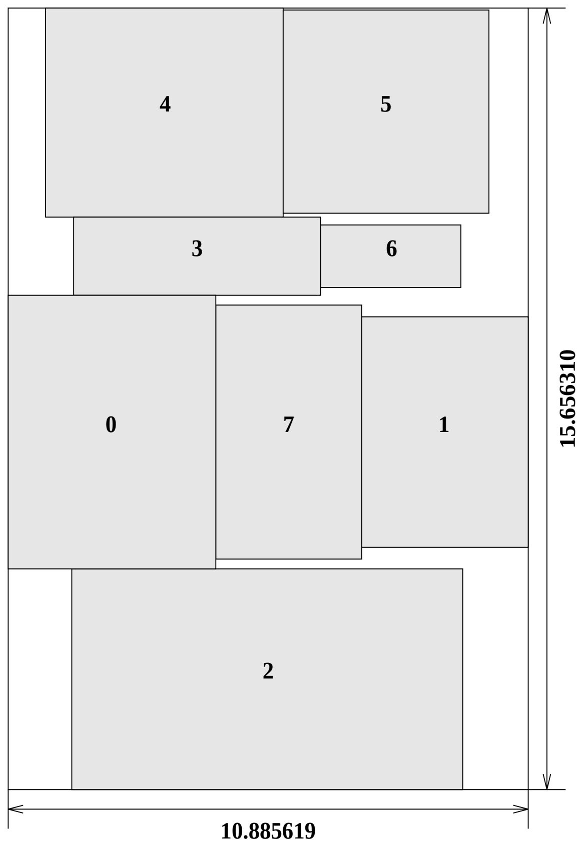


Figure G.8. The best layout for TL91-8

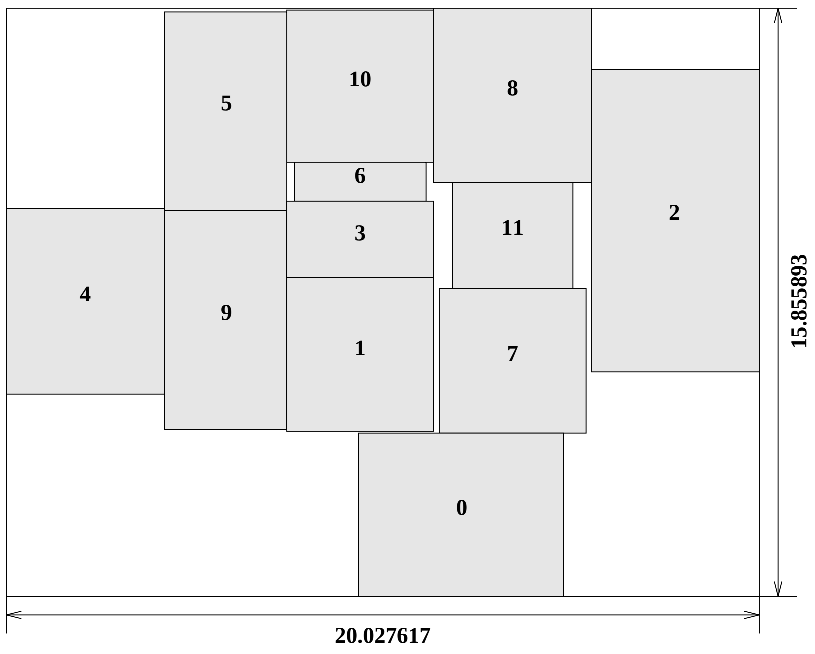


Figure G.9. The best layout for TL91-12

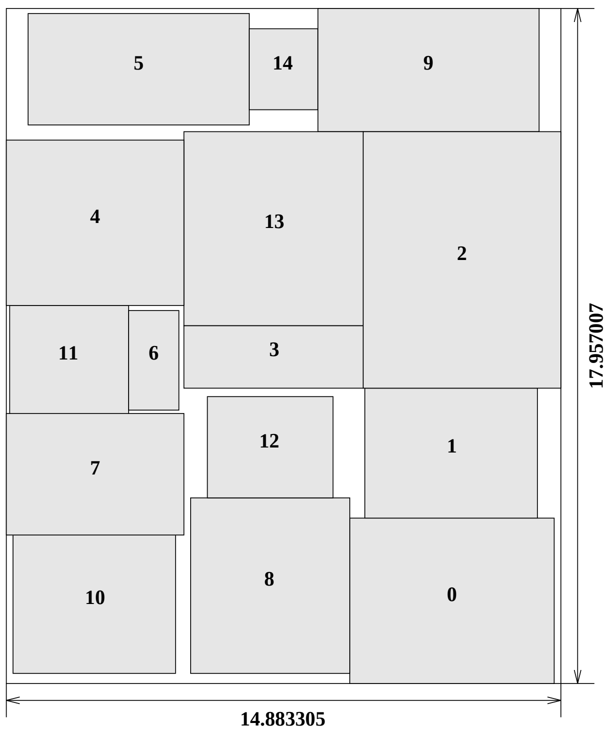


Figure G.10. The best layout for TL91-15

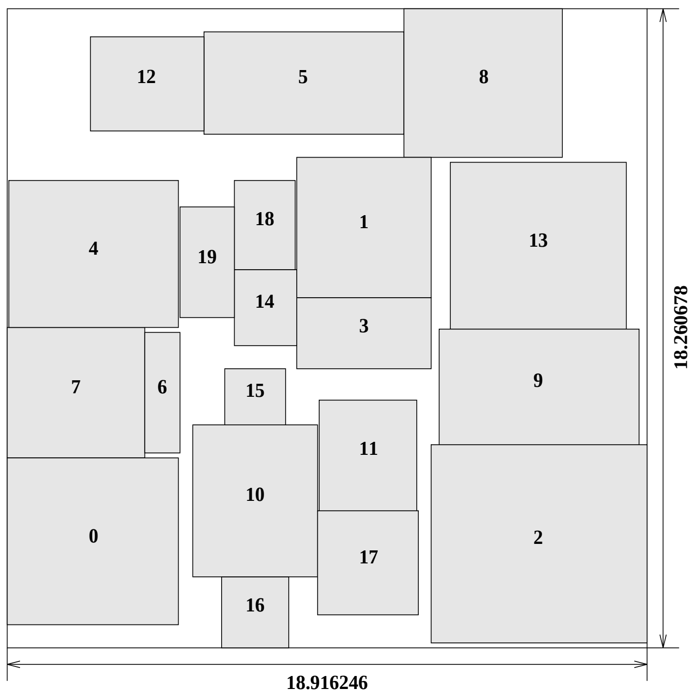


Figure G.11. The best layout for TL91-20

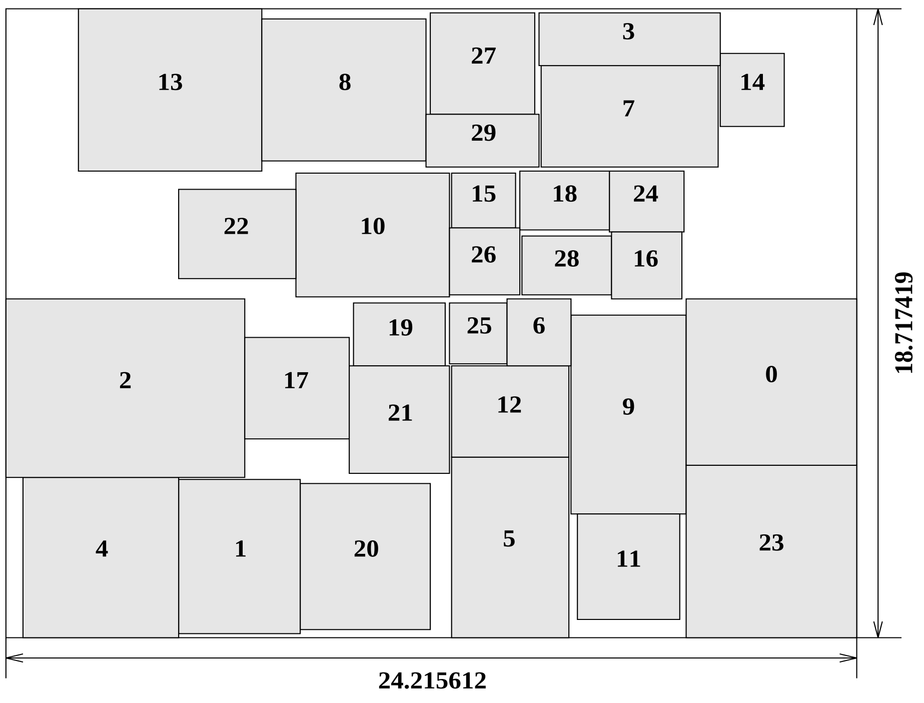


Figure G.12. The best layout for TL91-30

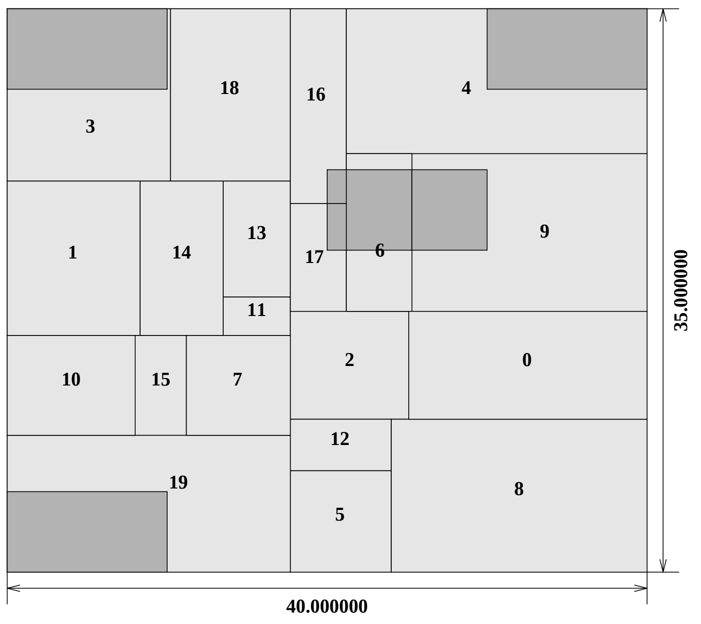


Figure G.13. The best layout for Tam92-20a

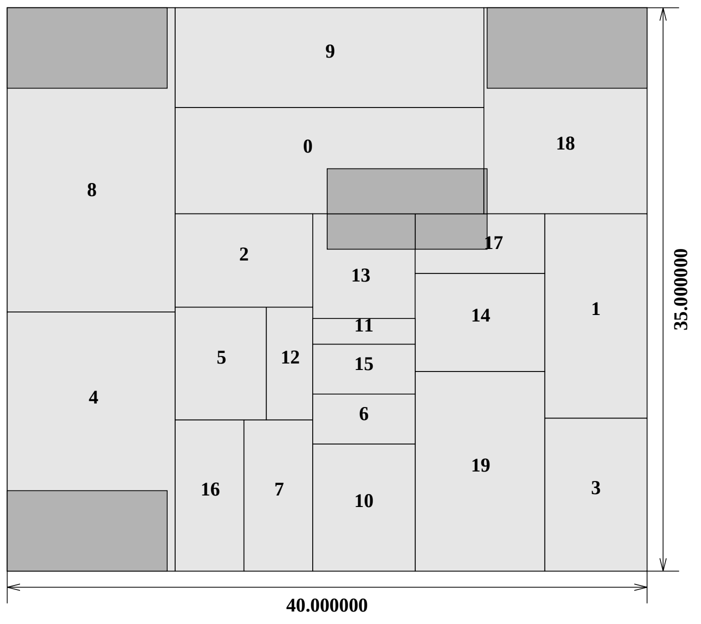


Figure G.14. The second best layout for Tam92-20a

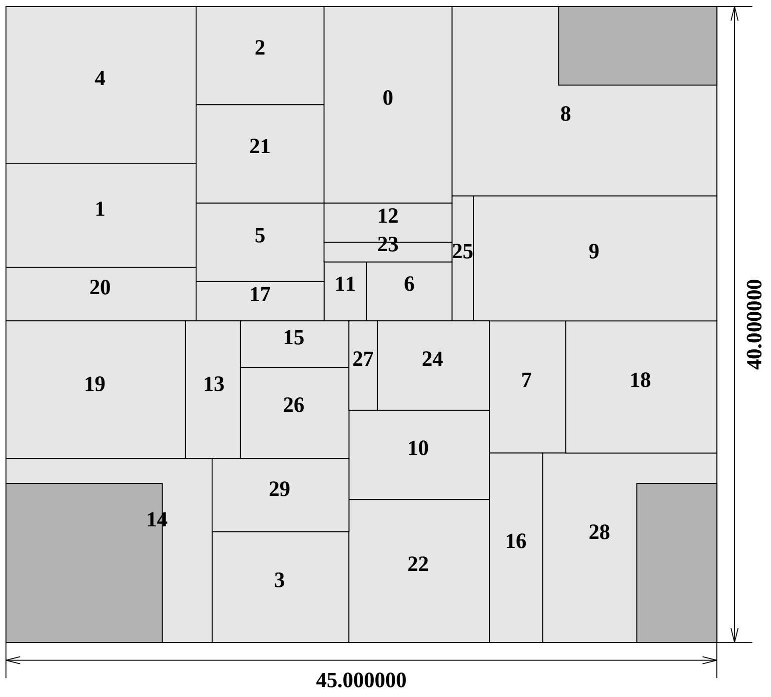


Figure G.15. The best layout for Tam92-30a

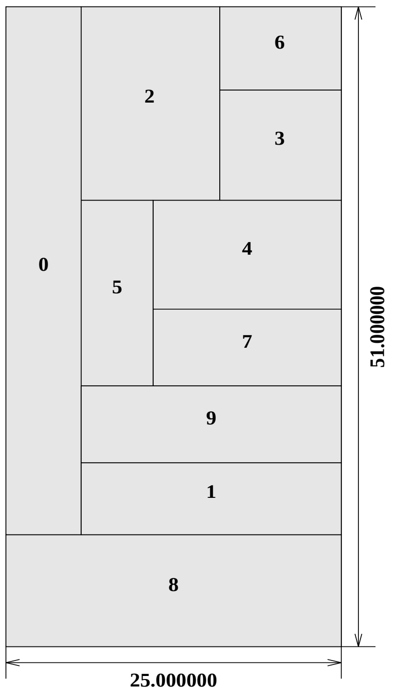


Figure G.16. The best layout for VCea91-10

```
-------------------------------------------------------------------------------------------------------------------------
problem      GA      score      chromosome
-------------------------------------------------------------------------------------------------------------------------

Kea91-11   twoTam2  2744.19  *-/+/**/+-72%8%14%%3%59%%0:%%6%
Kea91-11a  genCea   2180.07  908527*1*+4+*3*+6+*:*
Kea91-16   genTam     64.00  22022022022020
Kea91-20   genKad    158.00  +--*+//-+*+//+/++/+10%25%%63%79%%%8:%;?%>%%%4<%A%=B%@%C%%%
TL91-5     genKad    228.15  -*-/01%2%34%%
TL91-6     genKad    361.45  *--/-210%%34%5%%
TL91-7     genCea    595.89  210*63*54*++*
TL91-8     genCea    878.00  2071**36*+45*++
TL91-12    genCea   3283.12  4095+136+:++7;+8+**2*+*
TL91-15    twoKad   7383.99  ++/-*+/---/**+01%8<%%=3%2%%46;%%7:%%%5>%9%%
TL91-20    genKad  16392.52  *+--+//++-*-*++*/*+76%4%0%13%>B%C%%@:%?%;A%%%29%=%%5<%8%%
TL91-30    oneKad  41095.05  //*-*+-+*+*+/*+/*-*/-*/**-*-+D14%%2A%CE%%%9;%G0%%5<%I6%%%%>73%%=8%KM%%%?J%F:%%BH%L@%%%%%
Tam92-20a  genKad  20237.64  +/-*/-/--*/+*-**-+/A@%496%%%02%8<5%%%%C7:?%%%3B%1>=;%%%%%%
Tam92-20a  twoKad  20246.61  **++-+-/-+++/*+*++/@7%5<%%2%13%>A%C%%6:%?%;=%%%%09%B%%48%%
Tam92-30a  twoKad  43155.99  ++--+*/+*+-+/*-*/+*+/+-//-**-D14%%5A%2E%%%9I%8%;6%<0%G%%%%>M3%%=C%J?%%%HK%F:%%B7%L@%%%%%
VCea91-10  twoCea  19946.14  8019574+*236+*+++*+
-------------------------------------------------------------------------------------------------------------------------


        N.B.
        In the above table, "chromosome" indicates the chromosomes producing the best results.
        For convenience, "+-*/" and "0123" are used as operators of Polish expression instead of "UBRL"
        in the Cea/Tam2/DK2/Kad algorithms and in the Tam/DK algorithms, respectively;
        the facility index 0 to 29 correspond to "0123456789:;<=>?@ABCDEFGHIJKLM", respectively; and
        "%" is used to express the operator positions in the Polish expression template.
```

Figure G.17: Detailed information of GAs producing the best layouts