18 Application of Searching for Backbones to TSP

18.1 Definition of a Backbone

Now the searching for backbones (SfB) algorithm will be applied to the traveling salesman problem (TSP). As was already mentioned in Chap. 24 in Part I, one starts by creating a set of solutions and compares these for common parts. Figure 18.1 shows two quite good solutions for the PCB442 problem. At first glance, one finds the differences between these solutions. On the other hand, there are also common parts, namely, sequences of nodes, that are identical in both solutions.

Therefore, it is quite clear that this type of similarity should be used for the TSP for defining a backbone: as the TSP is a sequencing problem, this definition corresponds to the nature of the problem. Let us here now only consider the case of a symmetric TSP, i. e., $D(i, j) = D(j, i)$ for all pairs (i, j)

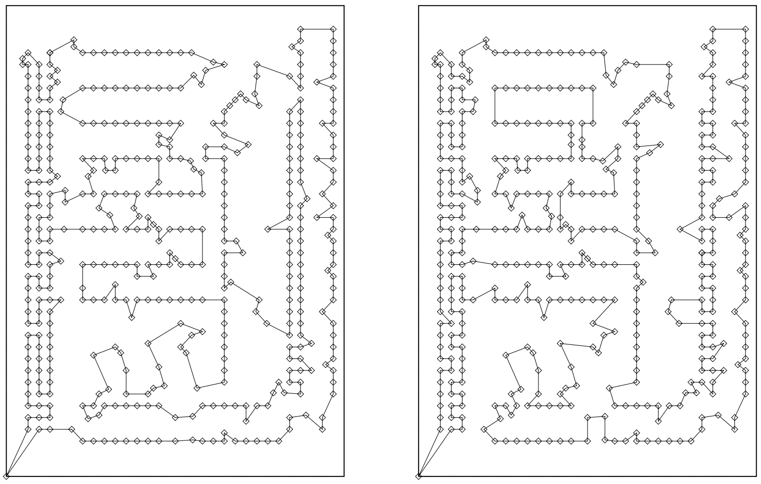

Fig. 18.1. Comparison of two quite good but not optimal solutions for the PCB442 problem. One finds many differences between these two solutions. However, there are also many sequences of nodes identical in both solutions

of nodes. Traversing each configuration either clockwise or counterclockwise leads to the same overall tour length.

The basic parts of the system are the individual nodes. The simplest relation between them is to ask whether they are neighbors of each other in one of the solutions. Let σ^ν be the solution of the run with number ν and let p be the number of compared solutions, which shall be kept constant here. Then the overlap matrix η_S for the symmetric TSP is defined as follows: the overlap between nodes i and j is given as

$$\eta_\mathrm{S}(i,j) = \sum_{\nu=1}^{p} \sum_{k=1}^{N} \delta_{i,\sigma^\nu(k)} \cdot \left(\delta_{j,\sigma^\nu(k-1)} + \delta_{j,\sigma^\nu(k+1)}\right) \,, \tag{18.1}$$

with $\sigma^\nu(0) \equiv \sigma^\nu(N)$ and $\sigma^\nu(1) \equiv \sigma^\nu(N+1)$. Therefore, one gets an overlap between two nodes i and j in the solution σ^ν if j is either the predecessor or the successor of i in the solution σ^ν. If $\eta_\mathrm{S}(i,j) = p$, then j is a neighbor of i in all solutions. In this case, they belong in one backbone. Using this overlap matrix, one can determine the backbones easily. A backbone is a sequence of nodes $i_1, i_2, \ldots, i_n$ with $\eta_\mathrm{S}(i_1,i_2) = \eta_\mathrm{S}(i_2,i_3) = \ldots = \eta_\mathrm{S}(i_{n-1},i_n) = p$. There is no node i_0 with $i_0 \neq i_2$ and $\eta_\mathrm{S}(i_0,i_1) = p$. Analogously, there is no such node i_{n+1} with $i_{n+1} \neq i_{n-1}$ and $\eta_\mathrm{S}(i_n,i_{n+1}) = p$.

After the backbones have been determined, they are supposed to be used for further optimization runs, in which they must not be destroyed. The simplest approach for achieving this is to ask in every move whether a chosen connection may be cut or not. This question can simply be answered by, e. g., looking at the corresponding entry in the overlap matrix η_S and checking whether it is maximal. However, this costs a large amount of calculation time. The more backbones one finds, the worse this situation is. Therefore, one must simplify this situation for subsequent optimization runs.

An obvious way to achieve this is to represent each backbone by a pair of nodes. Let us visualize this approach with a small example of a symmetric TSP instance with $N = 10$ nodes, for which the following four solutions were produced:

0 9 1 2 3 4 5 6 7 8
0 3 2 1 4 5 6 7 8 9
0 9 7 6 5 4 8 1 2 3
0 4 5 6 7 8 3 2 1 9

From these solutions one derives the overlap matrix

$$\eta_{\mathrm{S}} = \begin{pmatrix} 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 4 & 0 & 1 & 0 & 0 & 0 & 1 & 2 \\ 0 & 4 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 4 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 4 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 3 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 3 & 0 & 1 \\ 4 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} .$$

(Of course, the maximum entries here are identical to the number of solutions $p = 4$) and from that (using the first solution for splitting it into the backbones) the following set of backbones:

$$\begin{array}{l} \boxed{0 \quad 9} \\ \boxed{1 \quad 2 \quad 3} \\ \boxed{4 \quad 5 \quad 6 \quad 7} \\ \boxed{8} \end{array}$$

These backbones, which must not be destroyed in subsequent optimization runs, are now coded for the next iteration in such a way that the work for the processors is as easy as possible. One obvious way to do this is to code every backbone as a pair of nodes. In the next optimization runs, the tour may only be cut after pairs of nodes, i. e., after the second, fourth, sixth, etc. node. This can easily be achieved by producing only even integer random numbers.

Using this approach, all edge points of the backbones get a new number, and the nodes inside the backbones are removed. Therefore, the backbone set of the example above is coded in the following way:

$$\begin{array}{rcl} \boxed{0 \quad 9} & \Longrightarrow & \boxed{0 \quad 1} \\ \boxed{1 \quad 2 \quad 3} & \Longrightarrow & \boxed{2 \quad 3} \\ \boxed{4 \quad 5 \quad 6 \quad 7} & \Longrightarrow & \boxed{4 \quad 5} \\ \boxed{8} & \Longrightarrow & \boxed{6} \end{array}$$

One gets the following tour:

0–1 2–3 4–5 6–6

This tour is identical with the first solution of the previous SfB iteration. The "–" signs indicate that these connections must not be destroyed. Note that backbones containing only one node are doubled in the tour, such that this approach of choosing only even edges in the moves can be used.

Furthermore, one derives a smaller 7×7 distance matrix $\tilde{D}$ from the original 10×10 distance matrix D, which contains for example the following entries:

$$\widetilde{D}(0,1) = D(0,9), \widetilde{D}(2,3) = D(1,2) + D(2,3), \widetilde{D}(0,6) = D(0,8) .$$

This distance matrix and this tour are used for the next optimization runs. Of course, this coding never leads to a distance matrix $\tilde{D}$ with more entries than D. However, the tour could contain twice as many nodes as the original tour if the solutions are so different that no backbone containing at least two nodes can be found. If there are many backbones containing only one node, this coding costs additional time; on the other hand, it pays off if the solutions are rather similar to each other. Using either this or a related coding or the overlap matrix is necessary so as not to destroy the backbones.

In the second iteration of the algorithm, again a number of solutions is generated, but with a slightly altered program in which only edges after an even tour position number are allowed to be cut such that the backbones are not destroyed. Again the solutions shall be generated independently of each other. These new solutions must be decoded with the old backbones. Extending the example above, the new solutions could be

0–1 2–3 6–6 4–5
1–0 4–5 2–3 6–6
0–1 6–6 3–2 5–4
1–0 4–5 6–6 3–2

These solutions are decoded to:

0 9 1 2 3 8 4 5 6 7
9 0 4 5 6 7 1 2 3 8
0 9 8 3 2 1 7 6 5 4
9 0 4 5 6 7 8 3 2 1

As already described in Chap. 24 in Part I, these new solutions are supposed to be better than the previous ones and furthermore supposed to give a better representation of those parts of the problem instance that are already solved optimally. Therefore, the old solutions are discarded, but their inheritance consists of the backbones that were not allowed to be destroyed, such that they are also part of the new solutions. The new set of backbones is only constructed with the new solutions:

$$\boxed{0\quad 9}$$

$$\boxed{1\quad 2\quad 3\quad 8}$$

$$\boxed{4\quad 5\quad 6\quad 7}$$

With an increasing number of iterations, one gets fewer but longer backbones. The system to be optimized becomes smaller, and the calculation time per iteration therefore decreases. In this example, the tour now contains only six nodes and the distance matrix is of size 6×6.

18.2 Application to the Completely Asymmetric TSP

The asymmetric TSP (ATSP) exhibits a nonsymmetric distance matrix, i. e., there is at least one entry $D(i,j)$ with $D(i,j) \neq D(j,i)$. However, one must distinguish two cases for the ATSP: a completely asymmetric TSP has the property that for all pairs (i,j) of nodes $D(i,j) \neq D(j,i)$ and furthermore there is no sequence of nodes whose length stays the same if turned around. However, one usually finds the partially ATSP in practical applications in which there are some one-way streets in the problem instance by which only some percentage of distances becomes asymmetric.

The outline for the SfB algorithm for an ATSP is identical to that for the symmetric TSP; however, the definition of the overlap matrix, the determination and coding of the backbones, and the decoding of the coded solutions are different.

In the case of the completely asymmetric TSP, for which there is a preferable direction for any subsequence of nodes and for which there is no symmetry between clockwise and anticlockwise traversing the tour, the backbones must be determined in the following way: again the various solutions generated in the first iteration are compared with each other. However, only the traversion direction that is actually used in the individual solutions is used, i. e.,

$$\eta_{\mathrm{A}}(i,j) = \sum_{\nu=1}^{p} \sum_{k=1}^{N} \delta_{i,\sigma^\nu(k)} \cdot \delta_{j,\sigma^\nu(k+1)} \,, \tag{18.2}$$

with $\sigma^\nu(N+1) \equiv \sigma^\nu(1)$. This overlap matrix for the completely asymmetric TSP is therefore asymmetric. Then the backbones are determined according to the maximum entries, which again must have the value p. As these backbones are traversed in only one direction, it is sufficient to code them by only one node as they are not allowed to be either cut or turned around. This will be demonstrated in the following example. Let us consider a completely asymmetric TSP with $N = 10$ nodes, for which $p = 4$ solutions of the following form have been produced:

0 9 1 2 3 4 5 6 7 8
0 9 3 2 1 4 5 6 7 8
0 9 4 5 6 7 1 2 3 8
0 9 8 4 5 6 7 1 2 3

Then the overlap matrix is given by

$$\eta_{\mathrm{A}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} .$$

From this matrix the following backbone set is created:

$$\boxed{0 \quad 9}$$
$$\boxed{1}$$
$$\boxed{2}$$
$$\boxed{3}$$
$$\boxed{4 \quad 5 \quad 6 \quad 7}$$
$$\boxed{8}$$

These backbones are coded as follows:

$$\boxed{0}$$
$$\boxed{1}$$
$$\boxed{2}$$
$$\boxed{3}$$
$$\boxed{4}$$
$$\boxed{5}$$

From this one creates the new tour

$$0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \,,$$

which may be cut after each position, and a smaller 6×6 distance matrix $\tilde{D}$ with asymmetric distances between the various backbones, e. g.,

$$\tilde{D}(0,1) = D(9,1), \tilde{D}(1,0) = D(1,0), \tilde{D}(0,4) = D(9,4), \tilde{D}(4,0) = D(7,0) \,,$$

and furthermore distances inside the one-node backbones that have to be considered, e. g.,

$$\tilde{D}(0,0) = D(0,9), \tilde{D}(4,4) = D(4,5) + D(5,6) + D(6,7) \,.$$

In contrast to the symmetric TSP, the number of nodes in the tour, the number of nodes in the distance matrix, and the number of backbones are identical. Here also the diagonal elements of the distance matrix are used in order to calculate the length of a configuration. In conclusion, this coding already pays off if at least two nodes can be combined to one backbone when using a basic serial optimization algorithm that does not require a calculation of the total energy of the configuration when performing a change.

18.3 Application to Partially Asymmetric TSP

The coding is most difficult for the partially asymmetric TSP, as both symmetric and asymmetric backbones must be determined. Analogously to the two marginal cases of the completely symmetric and the completely asymmetric TSP above, first, the overlap matrices η_{S} and η_{A} are created. Then, again one of the solutions is chosen according to which the backbones are built. If $\eta_{\mathrm{S}}(i,j) = p$ for a pair (i,j) of nodes, then these nodes belong to one backbone. After the construction of the backbone set by using the symmetric overlap matrix η_{S} one must check, for all backbones containing more than one node, whether $\eta_{\mathrm{A}}(i_1, i_2) = p$ for the first two nodes in it. If this is the case, then automatically $\eta_{\mathrm{A}}(i_2, i_3) = \ldots = \eta_{\mathrm{A}}(i_{n-1}, i_n) = p$. In this case, the backbone must be marked as an asymmetric backbone; otherwise it is a symmetric backbone. Incidentally, if all backbones containing at least two nodes are marked as asymmetric, then the asymmetries in the distance matrix obviously dominate the system, such that one can proceed as in Sect. 18.2. Otherwise, one must represent each backbone by two nodes in the tour, as in the case of the symmetric TSP. However, the symmetric backbones are then coded with two different nodes, whereas the asymmetric backbones and the one-node backbones are represented by only one node, which is doubled in the tour. Again the diagonal elements of the distance matrix $\tilde{D}$ also must be added up when calculating the length of the solution.

At the end of the algorithm, all solutions will be identical, so that only one backbone containing all nodes is left. If the ground state of the problem is degenerate, a small number of backbones will remain at the end.

Generally, this algorithm is based on the idea that the backbones are considered to be optimal and, even more, to be part of the globally optimum solution. Of course, it is impossible to determine a priori which part belongs to an optimum solution. The assumption in this algorithm is that the statistical averaging over many good solutions will reproduce these backbones. This requires comparing an appropriate number of solutions. If there are too few solutions, then the backbones are too long already at the very beginning. Often they consist of nodes not connected in an optimum way. On the other hand, if there are too many solutions, then there are so many differences between them that pieces cannot be connected with each other as there is at least one solution voting for another possibility. The algorithm cannot converge in such a case. Therefore, the question of whether there is an optimum number of used solutions is of central importance.

We will present results for symmetric TSP instances. The calculations were performed on the massive parallel computer jump of the John von Neumann Institute for Computing at the Research Center Jülich, Germany. The IBM Fortran compiler mpxlf for the programming language Fortran 77 and the parallelization library MPI were used, working on $2^n, 1 \leq n \leq 7$ (2, 4, 8, 16, 32, 64, and 128) processors. Each processor created one solution in each SfB iteration.

Note that in all applications of the SfB algorithm, only the nearest-neighbor interaction between the nodes is considered, as described above: if two nodes are neighbors of each other in all solutions, then the link between them is added to the backbone list. Of course, this nearest-neighbor interaction is only a marginal case of the vast variety of correlations between various parts of a problem instance. However, due to the success this application has, it is sufficient to work with this nearest neighbor interaction only.

18.4 Computational Results

Now we investigate the results for an implementation of the SfB algorithm. We will concentrate here on the PCB442 instance, as this instance might be rather hard to solve for this algorithm due to the degeneracy of the ground state and of higher energy states of this instance. As the basic serial optimization algorithm used for providing solutions, we use simulated annealing (SA), as SA has the property that there are no restrictions in the search for good solutions due to the absence of constructive elements in SA. We always start with an initial temperature of 10^4 and cool the system down to a final temperature of 0.1 exponentially with a cooling factor of 0.99 and add a greedy step at the end. Thus, we used the same parameters for SA as in Sect. 7.4. Furthermore, each run starts with a random configuration, which is created by putting the nodes in a random order in the first iteration and the backbones in a random order in the other iterations. We use the node insertion move (NIM), the Lin-2-opt (L2O), and the four variants of the Lin-3-opt (L3O) with equal probability and leave out the exchange here. From the second iteration on, in which two-node-backbones instead of single nodes are used, the node insertion move becomes an edge insertion move: this move, which is also widely used for the standard TSP, shifts a pair of neighboring nodes to a new position. Please note again that also the moves used do not contain any constructive elements. The SfB algorithm ends either after a maximum of 1000 iterations or if less than three backbones are left, as at least three backbones are needed for performing a L3O or a NIM.

In [187] and [183], results for several observables describing the behavior of the SfB algorithm were already studied for several numbers p of compared solutions, but only for one fixed amount of calculation time. Thanks to the generous grant of computing time by the John von Neumann Institute, here we can study the quality of the algorithm both for various values of p ($p = 4$, 8, 16, 32, 64, and 128) and for various numbers of sweeps per temperature step in the SA algorithm (1, 3, 10, 30, 100, 300, 1000, and 3000 sweeps). If looking again at the graphics for the PCB442 instance in Fig. 7.9, we find that the quality of the results depends very strongly on the calculation time. For 10,000 and 30,000 sweeps per temperature step, we already obtained with some probability a ground state of the PCB442 instance with such a serial run such that we restrict ourselves now to shorter computing times for each SfB iteration. Furthermore, we must notice from Fig. 7.9 that rather different solutions are compared for these different computing times: for very short computing times, one compares rather bad solutions, which surely have much less in common with each other and with the global optimum, as there was no time to freeze these systems in locally minimum configurations, than those produced with larger amounts of computing time.

Thus, the question generally arises as to whether also for short computing times backbones can be found. To investigate this question we additionally performed some test SfB runs in which random configurations were compared for common parts. In the first iteration, the random configurations were created as usual; in the next iterations, the backbones were placed in a random order. If comparing only two random configurations in each iteration of the SfB algorithm for common parts, the algorithm converges to one random configuration within 1000 iterations. A small number of short backbones containing more than one node is even found in this time when using $p = 3$. For $p \geq 4$, no common structures in p different random configurations can be found within this time limit of 1000 iterations. Now we return to comparing solutions generated with SA.

Figure 18.2 shows the maximum number of nodes within a backbone for various computing times and numbers p of compared solutions. When working with $p = 128$, $p = 64$, $p = 32$, and $p = 16$, we find that the maximum number of nodes in a backbone remains 1 or close to 1 for short computing

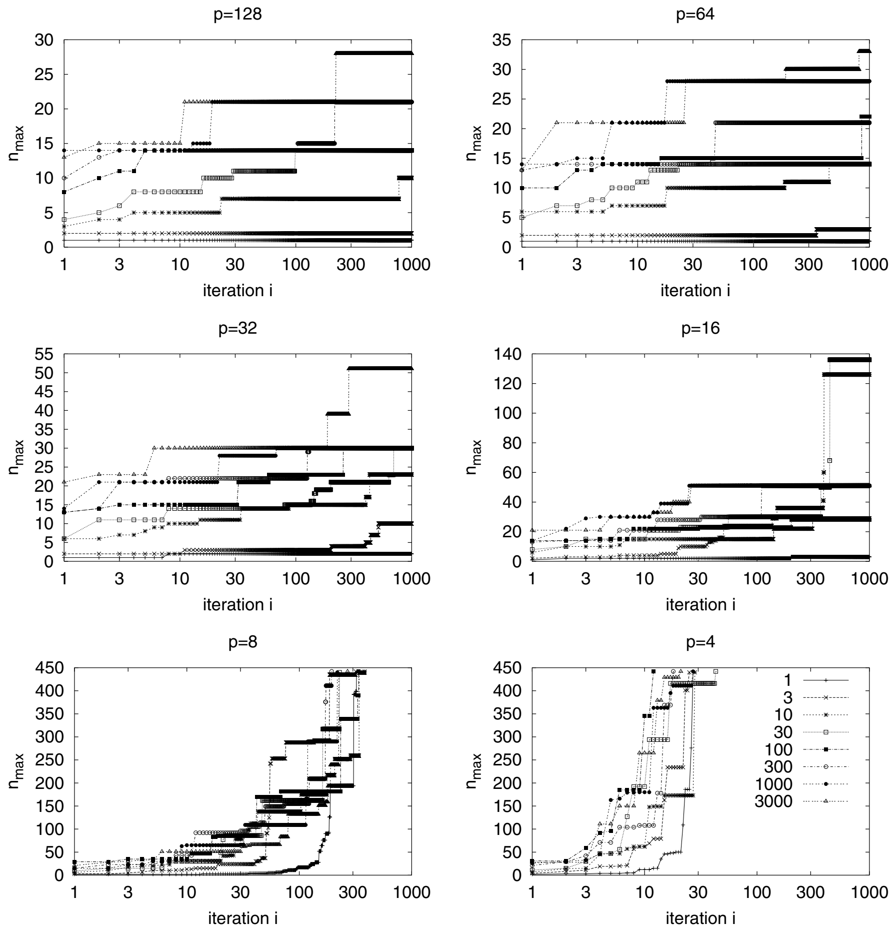

Fig. 18.2. Results for the application of SfB to the PCB442 instance for various numbers p ($p = 128, 64, 32, 16, 8, 4$) of compared solutions and for various calculation times. For the basic SA optimization algorithm, we use 1, 3, 10, 30, 100, 300, 1000, or 3000 sweeps per temperature step. These graphics show the maximum number of nodes $n_{\max}$ within a backbone vs. iteration i

times. Thus, either no backbone can be created or only very short ones. The solutions exhibit too many differences as the optimization processes hardly had any time to push the systems into local minima. If more computing time is invested, the maximum backbone size increases with an increasing number of iterations, but it never reaches the overall number of nodes in the system. Thus, ultimately the processors do not agree on a common solution. Contrarily, when working with $p = 8$ or $p = 4$, the maximum number of nodes in a backbone increases toward the system size within 1000 iterations for all calculation times. However, one cannot deduce from this behavior that the SfB algorithm would then always lead to the optimum solution. This behavior only shows that one can often find common structures if the number of solutions is appropriately small.

The relative mean deviation of the quality of the results achieved with the SfB algorithm to the optimum value of 50,783.5... of the PCB442 instance is shown in Fig. 18.3. For large numbers of processors, we get that this mean deviation, and therefore the average quality of the solutions, decreases only slightly with an increasing number of iterations. For short computing times,

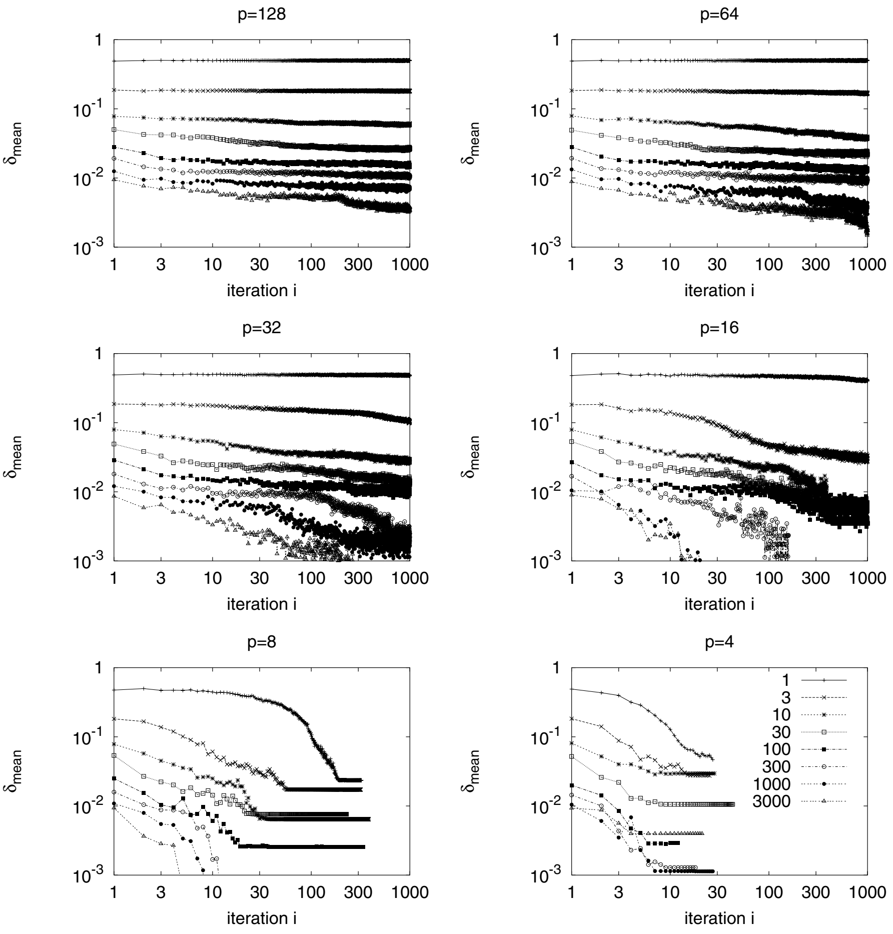

Fig. 18.3. Results of applying SfB to the PCB442 instance for various numbers p ($p = 128, 64, 32, 16, 8, 4$) of compared solutions and for various calculation times: for the basic SA optimization algorithm, we use 1, 3, 10, 30, 100, 300, 1000, or 3000 sweeps per temperature step. These graphics show the mean deviation of the results achieved in iteration i from the optimum

the average quality of the results is basically determined by the amount of calculation time invested. Given more time, some longer backbones could be created, such that obviously the SfB algorithm could lead to a decrease in the mean deviation with an increasing number of iterations. For $p = 32$ and $p = 16$, we find that the SfB algorithm does a really good job improving the average quality of the results if the serial calculation time is sufficiently long. For a small number of solutions, we also find strong improvements in the average quality of the solutions. However, the final average quality is also determined by the serial calculation time.

Figure 18.4 shows analogously the deviation of the best result found in an iteration from the optimum value for the PCB442 instance. Again we see that both the serial calculation time in each iteration and the number of processors have an influence on the results. Generally, one finds that if already a very good solution is found in some iteration, this does not necessarily mean that the best result of the next iteration will be either equally good or even better, as one might think because the backbones of the previous best solution are of course part of the new solutions. Thus, one should also always store the best solution found so far with the SfB algorithm, as the final solution might be worse.

Following the discussion about the improvement of the quality of the results by the SfB algorithm, we consider the convergence behavior of this algorithm. As the SfB algorithm unites several nodes to backbones, which will finally unite to only one backbone containing all nodes, it is useful to have a look at the number of backbones in the system, shown in Fig. 18.5. We find that this number gradually decreases with an increasing number of iterations. Thus, the system is always able to create new backbones consisting of a few former one-node backbones or to unite some longer backbones. However, only in the cases $p = 8$ and $p = 4$ is the algorithm able to converge to only one backbone within 1000 iterations.

Besides the number of backbones, which our coding routine makes equal to half the number of nodes in the tour, one can also investigate the convergence behavior by looking at the number of nodes in the distance matrix, which of course also decreases due to our coding as the system converges. This number is shown in Fig. 18.6. Like the number of backbones, the number of nodes in the distance matrix decreases gradually with an increasing number of iterations. Thus, the SfB algorithm also saves memory space, such that even for originally large TSP instances the distance matrix might finally fit in the cache, speeding up the calculation considerably.

This convergence behavior can best be described by the introduction of some order parameters for the algorithm. First, we want to define a parameter

$$\varphi(\eta_{\mathrm{S}}) = \frac{\sum\limits_{i,j} \delta_{\eta_{\mathrm{S}}(i,j),p}}{\alpha} \tag{18.3}$$

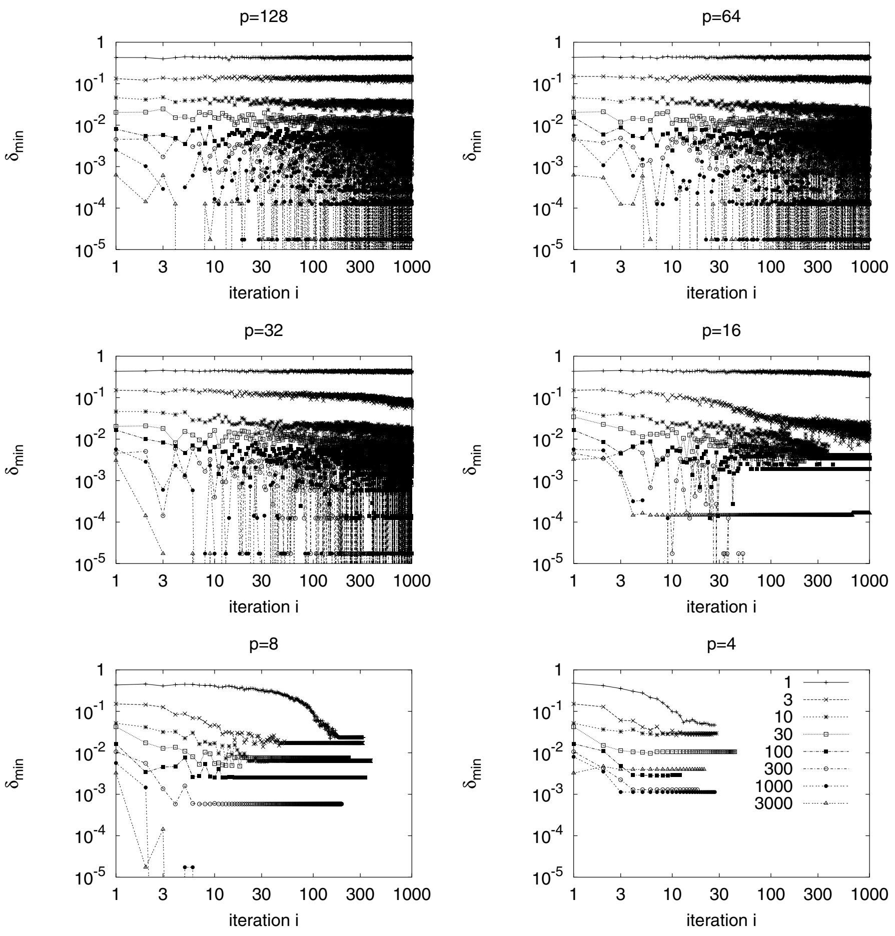

Fig. 18.4. Results of applying SfB to the PCB442 instance for various numbers p ($p = 128, 64, 32, 16, 8, 4$) of compared solutions and for various calculation times. For the basic SA optimization algorithm, we use 1, 3, 10, 30, 100, 300, 1000, or 3000 sweeps per temperature step. These graphics show the deviation of the best results in iteration i from the optimum

with $\alpha = 2N$ for the symmetric TSP. In the case of the completely asymmetric TSP, one sets $\alpha = N$ and uses η_A instead of η_S. If all solutions coincide, then two arbitrarily chosen nodes i and j are connected with each other in every solution, if they are connected in at least one solution, such that $\varphi = 1$. Contrarily, if there are so many differences between the solutions that no backbone consisting of at least two nodes can be created, then $\varphi = 0$. Figure 18.7 shows the results for this order parameter. We find that these graphics are like mirror images of the graphics for the number of nodes in the distance matrix shown in Fig. 18.6.

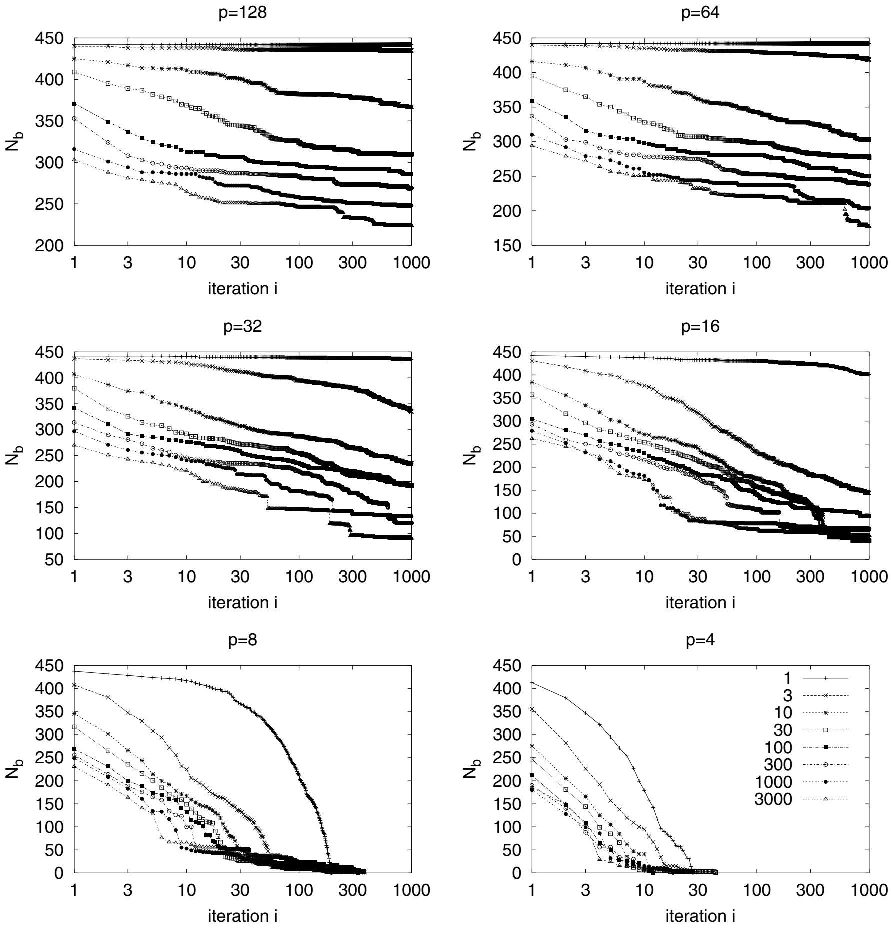

Fig. 18.5. Results of applying SfB to the PCB442 instance for various numbers p ($p = 128, 64, 32, 16, 8, 4$) of compared solutions and for various calculation times. For the basic SA optimization algorithm, we use 1, 3, 10, 30, 100, 300, 1000, or 3000 sweeps per temperature step. These graphics show the number of backbones, which is identical with half the number of nodes in the tour

A further order parameter that depends not on the maximum value of the entries in the edge matrix but on the number of zeroes in it can be defined as

$$\psi(\eta_{\mathrm{S}}) = 1 - \frac{-\alpha + \sum_{i,j}(1 - \delta_{\eta_{\mathrm{S}}(i,j),0})}{\alpha(p-1)} \,. \tag{18.4}$$

If all solutions are the same, then there are only $2N$ nonvanishing entries in the edge matrix of a symmetric TSP or N nonvanishing entries in the edge

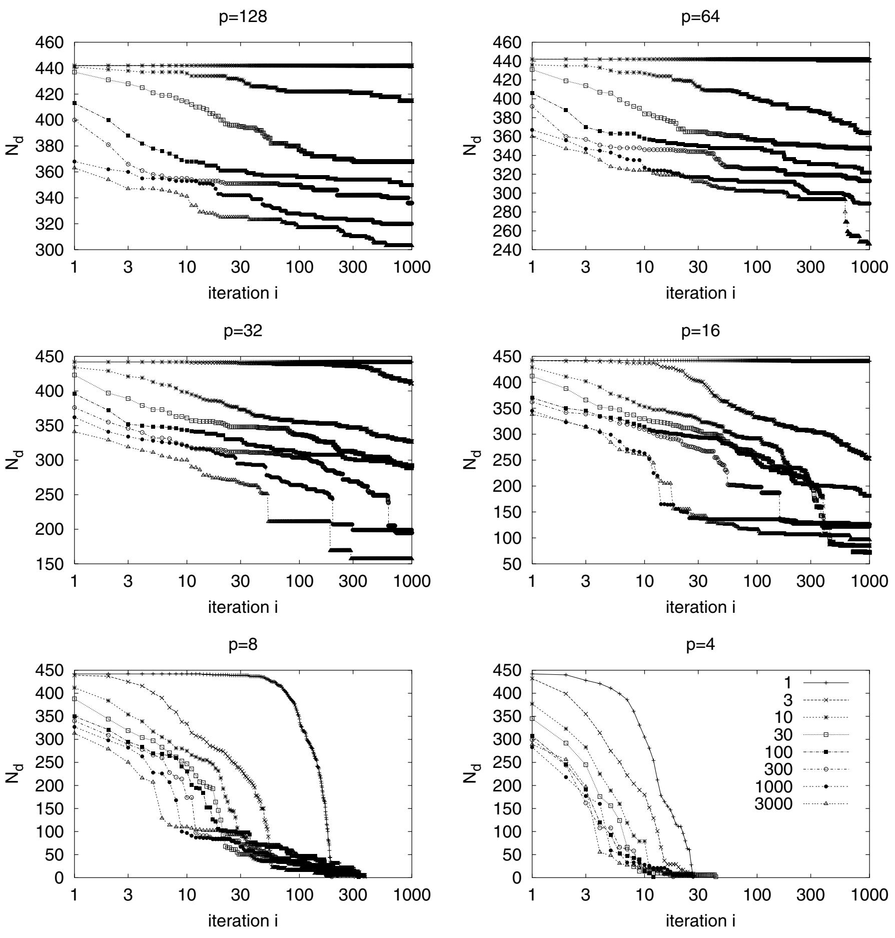

Fig. 18.6. Results of applying SfB to the PCB442 instance for various numbers p ($p = 128, 64, 32, 16, 8, 4$) of compared solutions and for various calculation times. For the basic SA optimization algorithm, we use 1, 3, 10, 30, 100, 300, 1000, or 3000 sweeps per temperature step. These graphics show the number of nodes in the distance matrix

matrix of an ATSP that are nonzero, thus $\psi = 1$. If all solutions are totally different (i. e., if node i is connected to node j in one solution, then these two nodes are not connected in any other solution), then there is a maximum number of nonzeroes in the edge matrix, namely, $2Np$ in the symmetric case and Np in the case of the ATSP, thus $\psi = 0$.

The results for the order parameter ψ are shown in Fig. 18.8. One sees at first glance that there are significant differences from the results for the order parameter φ in Fig. 18.7 for large values of p. In all cases, ψ is significantly

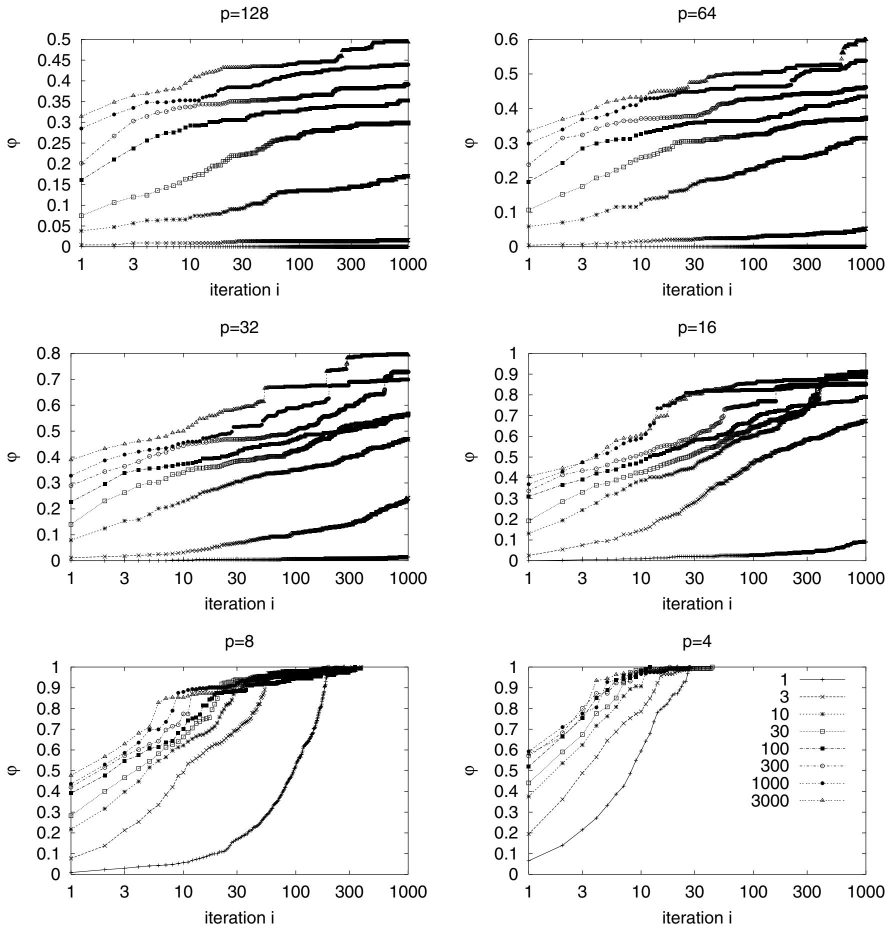

Fig. 18.7. Results of applying SfB to the PCB442 instance for various numbers p ($p = 128, 64, 32, 16, 8, 4$) of compared solutions and for various calculation times. For the basic SA optimization algorithm, we use 1, 3, 10, 30, 100, 300, 1000, or 3000 sweeps per temperature step. These graphics show the order parameter φ as defined in the text

larger than 0 already at the beginning. We get the smallest values for small p. For $p = 128$ and $p = 64$, ψ stays virtually constant, whereas ψ reaches a value of 1 for $p = 8$ and $p = 4$.

In the case $p = 2$, which is not shown here, φ and ψ generally coincide. For $p > 2$, one always gets $\psi > \varphi$. The difference between these parameters increases with an increasing number p. One can also provide a mathematical proof for this relation, which was done by Froschhammer and which is published in [187, 181, 182].

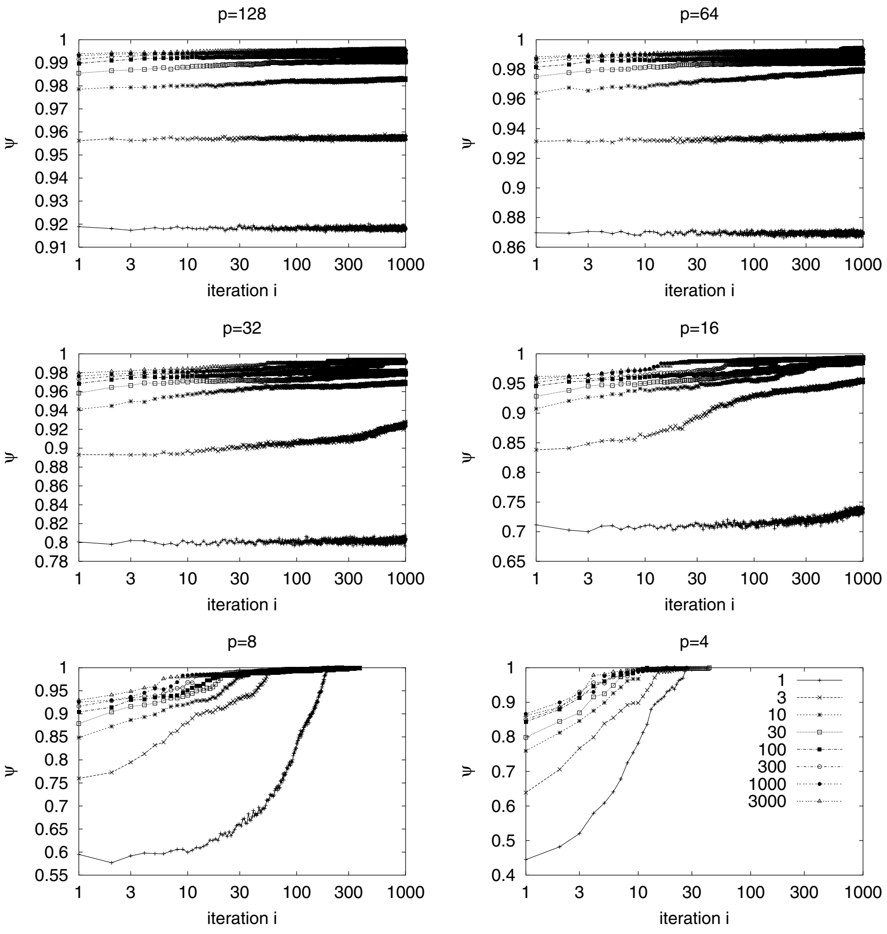

Fig. 18.8. Results of applying SfB to the PCB442 instance for various numbers p $(p = 128, 64, 32, 16, 8, 4)$ of compared solutions and for various calculation times. For the basic SA optimization algorithm, we use 1, 3, 10, 30, 100, 300, 1000, or 3000 sweeps per temperature step. These graphics show the order parameter ψ as defined in the text

Finally, we want to define a parameter for the overlap that measures the percentage of those nodes that have another specific node as predecessor or successor in all solutions. One can write this order parameter as

$$\xi = \frac{N - N_{\mathrm{b}}}{N} \, . \tag{18.5}$$

ξ vanishes if each backbone consists of one node only. This parameter is shown in Fig. 18.9. Of course, these curves are related to the curves in Fig. 18.5, which shows the number of backbones in the system.

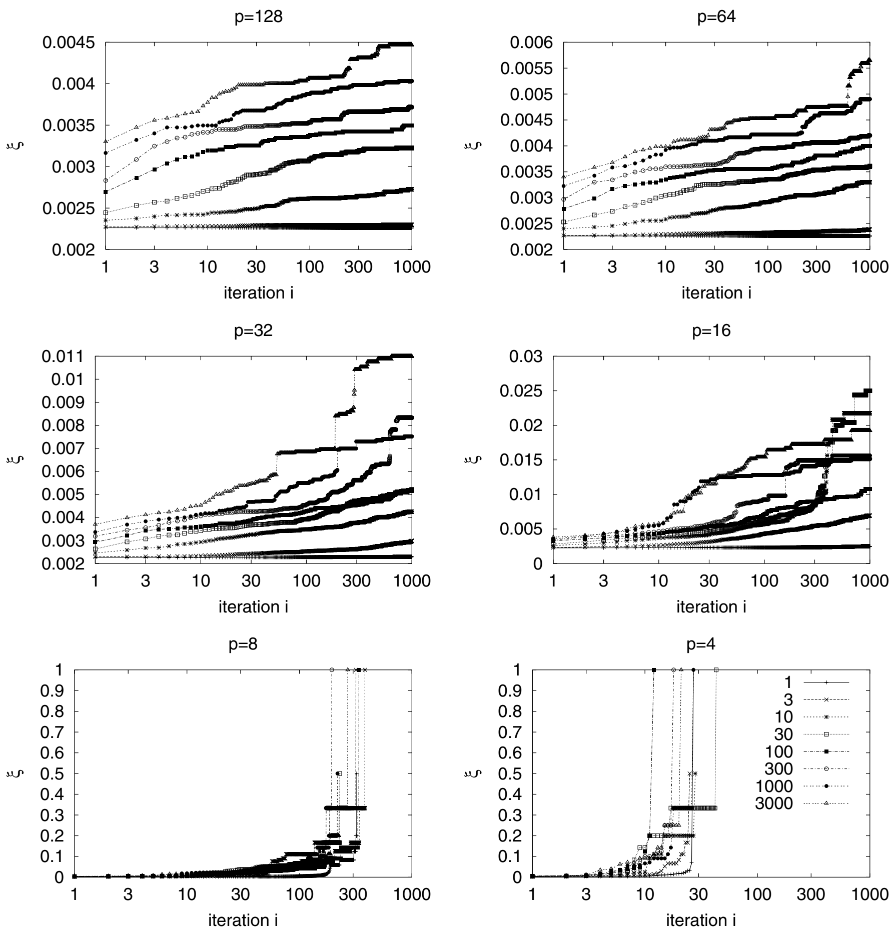

Fig. 18.9. Results of applying SfB to the PCB442 instance for various numbers p ($p = 128, 64, 32, 16, 8, 4$) of compared solutions and for various calculation times. For the basic SA optimization algorithm, we use 1, 3, 10, 30, 100, 300, 1000, or 3000 sweeps per temperature step. These graphics show the order parameter ξ as defined in the text