Searching for Backbones – A High-Performance Parallel Algorithm for Solving Combinatorial Optimization Problems

Johannes Schneider

Physik-Institut, Universität Zürich-Irchel, Winterthurerstr. 190, CH-8057 Zürich

Abstract

A highly efficient parallel algorithm called Searching for Backbones (SfB) is proposed: based on the fact, that many parts of a good configuration for a given optimization problem are the same in all other good solutions, SfB reduces the complexity of this problem by finding these "backbones" and eliminating them in order to get even better solutions in a very short time. Applications and results are presented for the Traveling Salesman Problem and the Vehicle Routing Problem.

Key words: Massive Parallelization, Optimization, Traveling Salesman Problem
PACS: 02.50.Ga, 05.10.Ln, 89.20.-a, 89.40.+k, 89.90.+n

1 Introduction

One of the main tasks in the fields of Computational Physics, Applied Mathematics, and Operations Research consists of finding a (quasi) optimum solution for a proposed problem according to a given energy/objective/cost function/Hamiltonian H. Many problems in this field belong to the class NP-complete, i.e., there are no analytical algorithms by which these complex problems can be solved in polynomial time. If the instance of the considered problem is of a large size, heuristic methods have to be used in order to achieve relatively good results or even the global optimum in a reasonable time. Many heuristics are constructive: starting from a tabula rasa, the

Email address: `Johannes.Schneider@physik.uni-regensburg.de` (Johannes Schneider).
URL: `http://rphibm1.physik.uni-regensburg.de/~scj04369` (Johannes Schneider).

various parts of the problem are introduced successively according to certain rules, until a complete solution, which fulfills all constraints, is received at the end. Alternatively, iterative improvement algorithms start from one or several randomly built (or already preoptimized) configurations and try to improve the already reached quality by a series of moves ($\sigma_i \rightarrow \sigma_{i+1}$), which change a configuration only slightly. Using Greedy-like algorithms, which only allow for improvements, often the optimization process soon ends up in a high-lying local minimum near the starting point in the energy landscape, failing much better configurations.

There are a few approaches to overcome this problem. Physical optimization algorithms introduce a temperature-like control parameter T in order to allow for some deteriorations and therefore to climb over barriers in the energy landscape. T is gradually reduced from a high value to zero. By that way, the optimization process is stepwise altered from the Random Walk mode, in which (nearly) every move is allowed, to the Greedy mode, in which every deterioration is forbidden. The system is transferred from a high-energetic unordered configuration to a low-energetic ordered solution. The various algorithms only differ in the exact choice of the transition rule: working with Simulated Annealing [1], mostly the Metropolis criterion [2] is used, which accepts every improvement; deteriorations are only allowed with a certain probability depending on the temperature T and on the energy difference $\Delta H = H(\sigma_{i+1}) - H(\sigma_i)$ between the actual configuration σ_i and the tentative new configuration σ_{i+1} [3]. In case of rejection, σ_{i+1} is set to σ_i. Threshold Accepting (TA) [4], an algorithm closely related to SA [5], uses a deterministic update rule [6]: TA accepts every move which does not worsen a configuration more than a certain threshold T.

Other improvement heuristics use e.g. Tabu principles (Already exploited parts of the energy landscape must not be visited again) or Darwin's "survival of the fittest" (Individua with a large fitness get more chances for reproduction, whereas bad individua have to commit suicide) [7].

2 Searching for Backbones

Comparing different good configurations for a given optimization problem, mostly many parts being equal in all of them can be found. Obviously, these "backbones" are already optimally solved. Each further serial optimization run wastes computing time in order to determine these backbones. This time could better be used for other parts of the problem, which are obviously more difficult to solve, such that better results could be found.

2.1 Modus Operandi

Searching for Backbones [8,9] is a parallel algorithm that makes use of the existence of backbones: it tries to find the global optimum or at least a very good configuration of the considered problem by a series of iterations. In the first iteration, a certain number p of independent optimization runs is performed, which produce a set of reasonable good solutions. All slave processors send their results to the master, which compares them and determines the equal parts. As these backbones are equally found in all different, independently generated configurations, they are assumed to be locally optimal.

Even more, the determined backbones shall be considered as part of the global optimum. From this idea, it follows that any further search for the optimum of the considered problem can be simplified: if the backbones (i.e. the easy parts of the problem) are held constant, the optimization process is able to concentrate on other parts, which are more difficult to solve. Hopefully, better solutions can be generated, which coincide in more equal parts than the previous configurations. Therefore, the strategy is as follows: the master returns the information about the backbones back to the slave processors. In this second iteration, the slaves independently perform new optimization runs, in which the already determined backbones must not be destroyed, and deliver their results again to the master. These new results are assumed to be of a higher quality than the previous ones and to provide a better representation of the already optimally solved parts of the problem. Therefore, the old configurations are neglected. However, their inheritance consists of the old backbone set, which is part of all new solutions. These are again compared for equal parts. Usually, they coincide in more parts, such that more and longer backbones are found. This procedure is repeated in further iterations:

- The master sends information about the backbones to the slaves.
- The slaves perform p independent optimization runs considering the backbones.
- The slaves send their results back to the master.
- The master determines a new set of backbones.
- If the various solutions still differ, goto step 1.
- The algorithm ends with an output of the final solution.

2.2 Application to the Traveling Salesman Problem

The application of this parallel algorithm on a given optimization problem can easily be demonstrated on the Traveling Salesman Problem (TSP): a TSP is given by a distance matrix D with $D(i,j)$ denoting the distance between the nodes i and j. The traveling salesperson has the task to find the shortest closed tour touching each of the N cities exactly once. Coding each configuration σ as a permutation of $(1 \dots N)$, the objective function H can be written as

$$H(\sigma) = D(\sigma(N), \sigma(1)) + \sum_{i=1}^{N-1} D(\sigma(i), \sigma(i+1)). \tag{1}$$

Many heuristics have been developed for finding a good solution for this NP-complete problem [10]. If such a heuristic contains enough random choices to sample the whole configuration space for good solutions (It is not sufficient to produce at least some slightly different results), then it can be used as a basic serial optimization algorithm for SfB. Physical optimization algorithms like TA, which shall be considered throughout this paper, meet this constraint.

Traditionally, the Swap has been used as move for the TSP. It exchanges two nodes in the tour. Lin introduced the Lin-2-Opt (L2O), which cuts two connections and turns of the two resulting parts of the tour around. The L2O leads to better results than the Swap [11]. Additionally, one variant of the Lin-3-Opt (L3O) which eachanges two succeeding parts of the tour without altering their directions shall be used as move, since the L3O leads to better results than the L2O [11]. According to my experience, using both the L2O and this L3O leads to even better results [8,5].

Let us consider a small example of a symmetric (i.e. $D(i,j) = D(j,i)\ \forall\, (i,j)$) TSP with $N = 10$ nodes and $p = 4$ processors. The four solutions shall be:

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

Comparing these solutions, the master creates a symmetric overlap matrix

$$\eta_S(i,j) = \sum_{\nu=1}^{p} \sum_{k=1}^{N} \delta_{i,\sigma^\nu(k)} \cdot \left(\delta_{j,\sigma^\nu(k-1)} + \delta_{j,\sigma^\nu(k+1)} \right) \tag{2}$$

(with $\sigma^\nu(0) \equiv \sigma^\nu(N)$ and $\sigma^\nu(1) \equiv \sigma^\nu(N+1)$) between the single nodes, i.e., one gets an overlap according to configuation σ^ν between the nodes i and j, if j is either predecessor or successor of i in the solution σ^ν. If $\eta_S(i,j) = p$, then j is neighbored to i in all solutions. It follows that i and j belong to the same backbone in this case. For the example above, the overlap matrix

$$\eta_S = \begin{pmatrix} 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 4 & 0 & 1 & 0 & 0 & 0 & 1 & 2 \\ 0 & 4 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 4 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 4 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 3 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 3 & 0 & 1 \\ 4 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

and the backbone set

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

are retrieved. These backbones have to be coded for the next iterations, such that the verification whether a backbone is destroyed is as simple and time saving as possible. One way to achieve this is coding each backbone by a pair of nodes:

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

Backbones consisting of only one node are put twice in the tour, such that the following tour is returned to the slaves:

$$0-1 \quad 2-3 \quad 4-5 \quad 6-6$$

The "-" symbols indicate that these odd connections must not be destroyed, the slaves are only allowed to cut the tour after the 2nd, the 4th, the 6th, ... node, which is very simple to implement. Additionally, a 7×7 distance matrix $\widetilde{D}$ is returned to the slaves, which consists of e.g. the following entries:

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

The slaves perform independently their optimization runs and deliver their results back to the master, e.g.:

$0-1\ 2-3\ 6-6\ 4-5$
$1-0\ 4-5\ 2-3\ 6-6$
$0-1\ 6-6\ 3-2\ 5-4$
$1-0\ 4-5\ 6-6\ 3-2$

The master uses the old backbone set in order to decode the solutions of the slaves:

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

One finds that the new backbone set is smaller than the previous one and proceeds with the algorithm until a final common solution is found.

2.3 Discussion of the Algorithm

Searching for Backbones can thus be considered as an algorithm, which gradually reduces the complexity of a problem. The definition, determination and coding of backbones is strongly problem dependent, e.g., it is sufficient to code each backbone by only one node if the TSP is totally asymmetric [5]. However, the definition of an item which shall be identified as a backbone is mostly relatively straightforward. Often the definition can be adopted from a simpler problem, which serves as a substructure of the considered problem, e.g. the considerations made for the TSP can simply be transferred to Vehicle Routing Problems.

The problem of determing a proper value for p has to be solved for every new problem: if p is too large, the solutions often exhibit too many differences, such that (nearly) no backbones can be built. However, a too small value of p would lead to false assumptions: false backbones could be built because some parts of the configuration space are not represented in the set of solutions. p has to be large enough to provide a good statistical significance. Therefore, there must exist a medium sized optimum value for p. Alternatively, backbones can be built from the $\widetilde{p}$ best solutions out of p results, if one wants to work with a large p but faces the problem that no convergence can be achieved with this p.

3 Computational Results for the TSP

Reinelt collected several TSP instances in his library TSPLIB95 [12]. For many instances, like the problem of the 127 beergardens in the area of Augsburg, the optimum solution is already found in the first iteration of the SfB algorithm by the basic serial optimization run. Therefore, a more difficult TSP instance shall be considered, the PCB442 problem consisting of $N = 442$ drilling holes on a curcuit, which was introduced by Grötschel and mathematically exactly solved by Holland [13], who found one optimum solution with $H_0 = 50783.5475\ldots$. However, this instance has a highly degenerate ground state, a fact which leads to large difficulties for many optimization algorithms. The SfB algorithm is expected not to converge either for medium or for large p.

SfB and the underlying serial optimization algorithm TA are implemented in Fortran 77. The message passing system Message Passing Interface is used. The program was run on Intel Paragons with queues for 2^n $(1 \leq n \leq 7)$ processors; each of them produces exactly one solution, which are sent to the master processor. The algorithm stops if their energy values are identical or if 1000 iterations are performed at maximum.

3.1 Improvement of the Results

First of all, the improvement of the results shall be investigated. Figure 1 shows the decrease of the energy with an increasing number of iterations for different processor numbers $p = 2^n$. The first iteration produces solutions analogously to simple serial optimization runs. Using a small p, all energies converge against a common value. At large values of p, the minimum and maximum lengths show large fluctuations between successive iterations, whereas the mean value fluctuates much less. It shows a significant decrease, especially in the first 10 iterations.

3.2 Convergence of the Algorithm

Secondly, the convergence of the algorithm is very interesting especially for the considered PCB442 problem due to its degenerate ground state. This can be best seen at the number of the nodes in the distance matrix and in the tour. Their decrease is shown for different p in Fig. 2. Of course, their number is 442 in the first iteration. Due to the coding explained in section 2.2 the number of nodes in the tour can be up to twice as large as 442 if the single solutions differ so much that many backbones consist only of one node. It increases between the first and the second iteration for $p > 2$. After that, it decreases

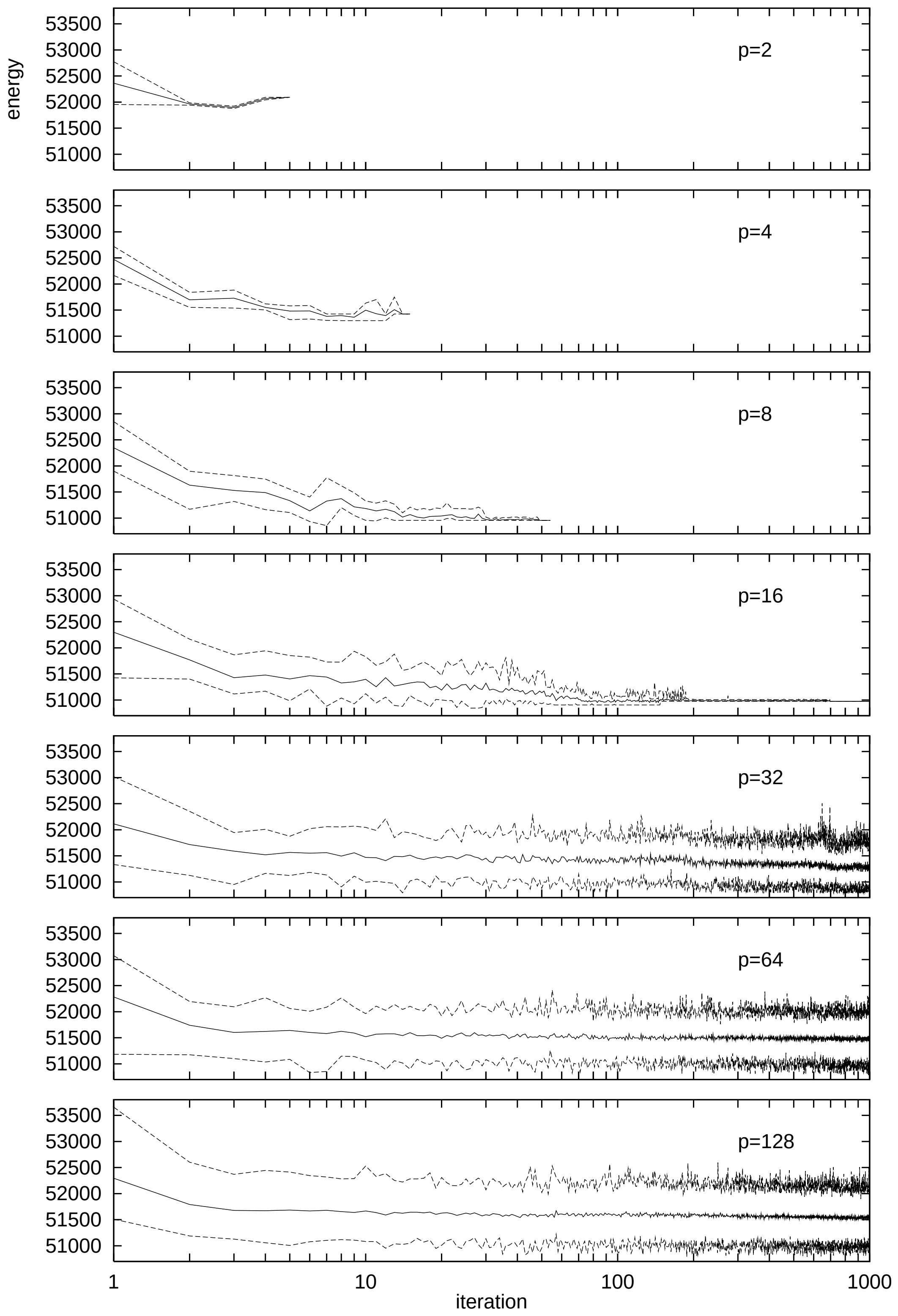

Fig. 1. Decrease of the minimum, mean, and maximum energies of the PCB442 problem with an increasing number of iterations, if working with Searching for Backbones on different processor numbers p.

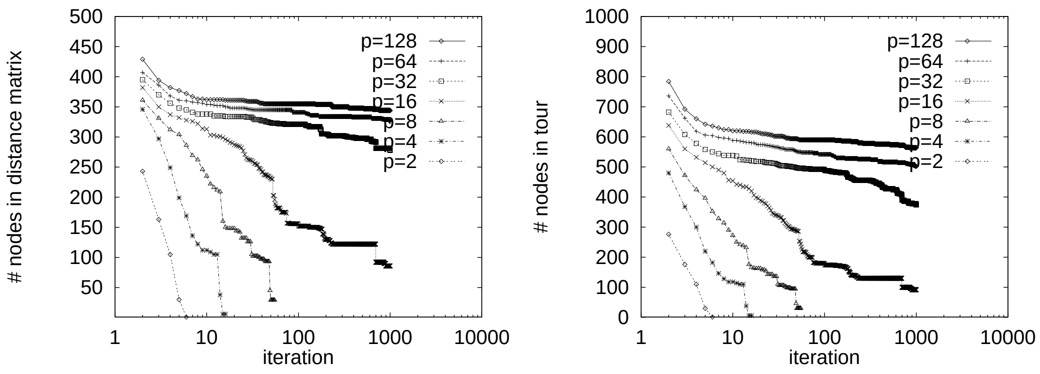

Fig. 2. Development of the number of nodes in the distance matrix (left) and in the tour (right) with increasing number of iterations for different processor numbers p.

monotonically. With an increasing number of iterations, this decrease becomes gradually, only seldomly backbones can be unified. Obviously, there are some small parts which are easily solved in the first iterations. However, it takes many iterations to get further commonness between the different solutions and to be thus able to unify short backbones to longer ones.

Other observables can easily be derived from the number of nodes in the tour: the number of the backbones is half as large as the number of nodes. The mean backbone length is given as the ratio between the original number of points (e.g. 442) and the number of backbones. It stays between 1 and 2 for large p and slowly increases for medium sized p ($8 \leq p \leq 32$).

Obviously, there are some nodes in this PCB442 problem, which cannot be connected with certain other nodes in a nontrivial way. Using $p > 4$, the probability that all solutions locally vote for the same possibility is very small, such that the minimum backbone length stays 1 [8].

3.3 Further Results

Usually, a speedup is considered if working on a parallel computer. It is defined as $S(p) = C_1/C_p$ with C_i denoting the calculation time on i processors. This definiton, however, is only suitable if the solution process for a problem can simply be divided between different processors. Working with Searching for Backbones, other definitions must be found. Remembering the statement that SfB reduces the complexity of a problem, observables which describe this complexity and influence the calculation time in each iteration have to be investigated.

The first approach leading to such observables are the numbers of the remaining nodes in the distance matrix and in the tour, which were already shown in Fig. 2. These numbers strongly influence the calculation time per iteration.

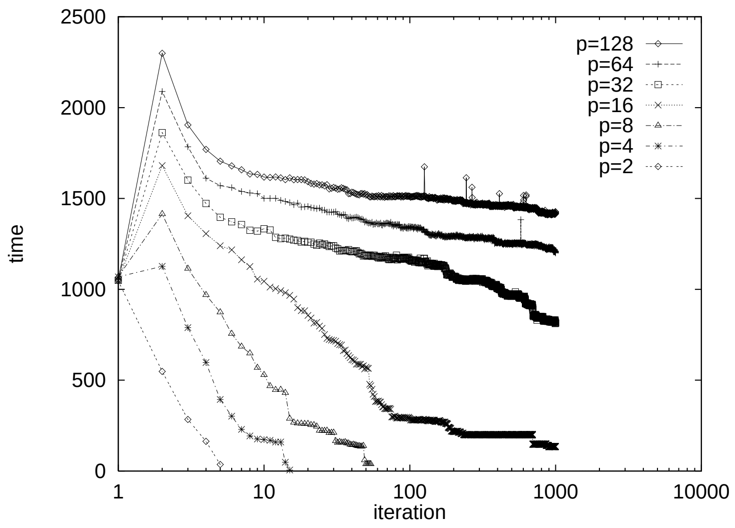

Fig. 3. Decrease of the calculation time spent in each iteration for different processor numbers p.

Figure 3 shows the calculation time for each iteration from the time when the master starts to send its data to the slaves to the moment in which it receives all new results. Due to the coding, the calculation time increases between the first and second iteration for $p \geq 4$. After that, it decreases monotonically.

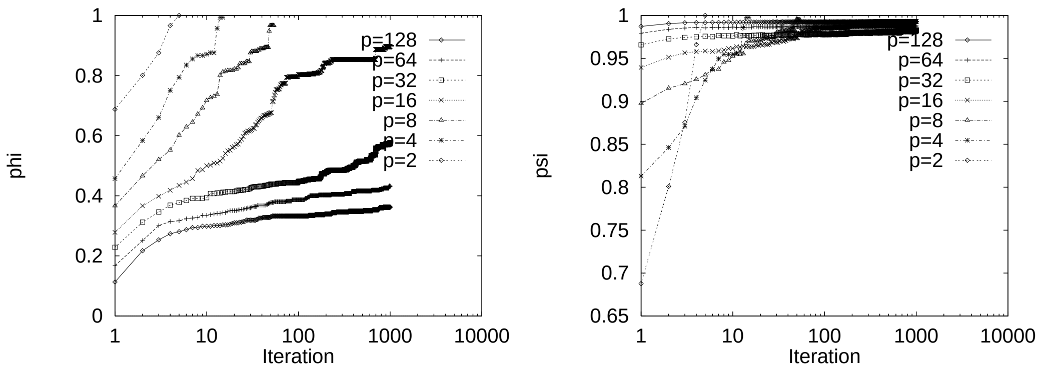

Fig. 4. Order parameters φ (left) and ψ (right) for different processor numbers p.

Upon these reflections, a parameter φ which measures to which extent the system is already solved [8,9] shall be considered:

$$\varphi(\eta_S) = \frac{\sum\limits_{i,j} \delta_{\eta_S(i,j),p}}{2N} \tag{3}$$

with the Kronecker symbol δ. It was shown that φ is a good order parameter reflecting the remaining complexity of a problem [14]. If all solutions are the same, the nodes i and j are connected in every solution if they are neighbored in at least one solution, such that $\varphi = 1$. If there are so many differences between the various solutions that no backbone can be built, then $\varphi = 0$. The curves in Fig. 4 for φ are similar to those in Fig. 2, they simply seem to be only mirrored horizontally. Already at the beginning, more than 15% of the nodes have the same point as predecessor or successor in all solutions for all considered p. The problem is even more simplified in the following iterations. Due to the large degeneracy of the ground state and thus also of other local optima, φ cannot reach a value of 1 for large p.

A further order parameter, which measures the number of zeros in the edge matrix, is defined by

$$\psi(\eta_S) = 1 - \frac{-2N + \sum_{i,j}(1 - \delta_{\eta_S(i,j),0})}{2N(p-1)}. \tag{4}$$

If all solutions are identical, then $\psi = 1$, if they are totally different, then $\psi = 0$. Of course, φ is equal to ψ for $p = 2$ (for the proof see [9]); for larger p, φ is always smaller than ψ (see Fig. 4) as expected.

4 Solution of the PCB442 Problem

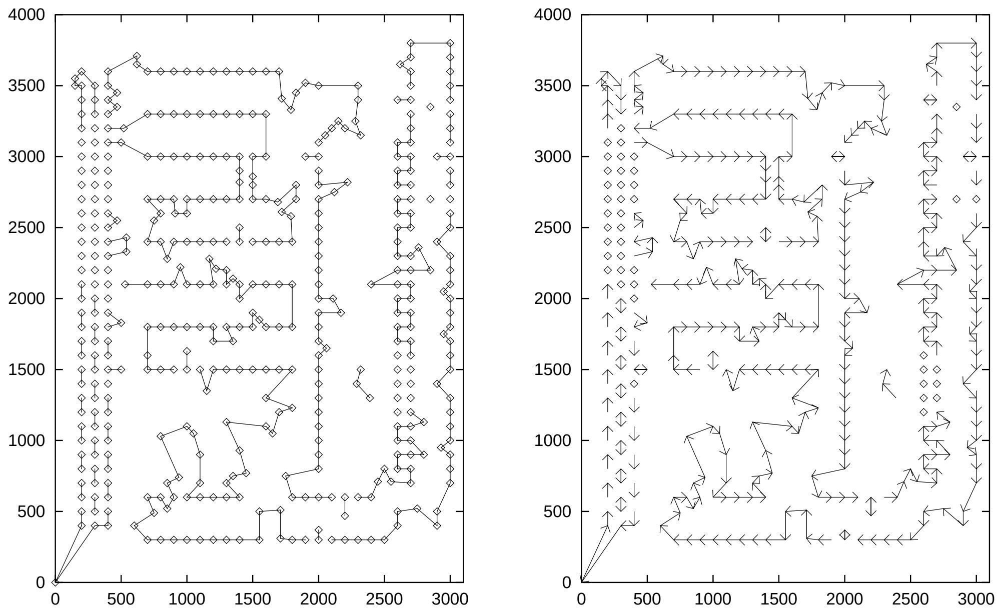

Fig. 5. The 89 backbones of the PCB442 problem found by comparing over one million optimum solutions (left); Super backbone of the PCB442 problem (right): fixing the direction of an arbitrary backbone consisting of at least three nodes, the directions of all other backbones except 16 blinkers (which consist of two nodes each) are determined.

Using the 32 best solutions of 128 independently produced configurations in the SfB algorithm, more than one million optimum solutions with minimum energy could be found. Comparing all these solutions, 89 backbones can be identified, which are shown in Fig. 5. However, these 89 backbones are not independent of each other. They are only cut at some positions where a local degeneracy exists, such that there are at least two possibilities to connect them optimally. If the direction of a backbone which consists of at least three nodes is fixed, one finds that the directions of all other backbones are also fixed except 16 blinkers, which consist of two nodes each. Therefore, the PCB442 problem introduced by Grötschel consists of a super backbone of 369 nodes, 16 blinkers of length 2, and 41 single nodes. This local determination of the overall direction is a speciality of the PCB442 problem, since counterexamples can simply be found [5].

5 Computational Results for the VRP

The Vehicle Routing Problem consists of a basic Multi Traveling Salesmen Problem, i.e. several salespersons deliver/fetch goods of a certain size/amount m to/from n customers, with the additional constraint that their trucks have only a finite capacity. Usually, the case is considered that the k salespersons start and end their tour at the same node, which is called depot, and that all trucks have the same capacity κ. A configuration is therefore given by a two dimensional array σ with $\sigma(i,l)$ denoting the i th customer in the tour of the l th traveling salesperson.

The concept of penalty functions is well suited to handle constraints: if a configuration is not feasible due to at least one overloaded truck, then it is not forbidden during the optimization process. It is only punished by enlarging its energy, such that hopefully the optimization process, which searches for configurations with minimum energy, ends in a feasible solution. Each constraint is mapped on such a penalty energy, which is simply added to the Hamiltonian. These considerations yield to the Hamiltonian [5]

$$\begin{aligned} H(\sigma) = {} & \sum_{l=1}^{k} \sum_{i=1}^{N_l-1} D(\sigma(i,l), \sigma(i+1,l)) \\ & + \lambda \sum_{l=1}^{k} \left(\sum_{i=2}^{N_l-1} m(\sigma(i,l)) - \kappa + \gamma \right) \Theta \left(\sum_{i=2}^{N_l-1} m(\sigma(i,l)) - \kappa \right) \end{aligned} \tag{5}$$

with an additional offset γ, the Lagrange multiplier λ, and the Heaviside function $\Theta(x) = 1$ if $x > 0$ and 0 otherwise. There are N_l customers in tour l. The first addend in equation (5) is the conventional TSP part summarizing over the edges used in the configuration σ. The second addend weights the overloadings of the single trucks l with a factor $\lambda > 0$. The Heaviside function is needed to punish only overloadings.

Now the question remains how to define a backbone. There are two obvious choices reflecting general properties of the VRP:

- Considering the VRP as a distribution problem (i.e. the customers are distributed on the tours of the various trucks), one can put two customers together in a backbone if they are in the same tour in all solutions.
- Alternatively, the VRP can be viewed as a sequencing problem. Analogously to the TSP, an edge matrix

$$\eta_S(i,j) = \sum_{\nu=1}^{p} \sum_{l=1}^{k} \sum_{q=2}^{N_l-1} \delta_{i,\sigma^\nu(q,l)} \cdot \left(\delta_{j,\sigma^\nu(q-1,l)} + \delta_{j,\sigma^\nu(q+1,l)}\right) \tag{6}$$

  $(i, j > 1)$ can be built, which considers the sequence of customers in the tour but neglects the connections to the depot (node No. 1).

Working with this second definition on $p = 32$ processors, the Searching for Backbones algorithm was able to reproduce the best results and to produce even new world records [5] for all instances in a collection of libraries composed by P. Augerat [15]. The new values are given in table 1. The results provided for the PLIB in [15] are shady: a verification for the small P-n16-k8 instance by a run with an exact Branch & Bound algorithm led to an optimum value of 450, which is larger than the value of 435 given in the literature. Therefore, a comparison was not drawn for the PLIB.

6 Conclusion

I presented a very powerful algorithm for solving combinatorial optimization problems: Searching for Backbones tries to find equal parts in different solutions for a given optimization problem. These backbones are assumed to be already optimally solved and are held constant during the remaining part of the optimization process, such that the complexity of the problem is reduced and even better solutions can be found. The strength of this algorithm was demonstrated for the Traveling Salesman Problem and the Vehicle Routing Problem. The ansatz described in this paper for the TSP can simply be transferred on any other sequencing problem, e.g., good results are achieved for supply chain problems [16]. The algorithm can also be extended to other types of problems; I will continue the research on problems with binary variables, such as satisfiability problems and error correcting codes, in which also structures which are common to all solutions can be found.

Table 1
New optimum values found by the Searching for Backbones algorithm for instances of the ALIB, BLIB, COUNTRYLIB, EILLIB, FISHERLIB, and MLIB. The values from the literature [15] are given for comparison.

ALIB		
Problem	Lit.	SfB
A-n39-k5	825	822
A-n39-k6	833	831
A-n45-k6	948	944
A-n60-k9	1408	1354
A-n61-k9	1035	1034
A-n62-k8	1290	1288
A-n63-k9	1634	1616
A-n63-k10	1315	1314
A-n64-k9	1402	1401
A-n65-k9	1177	1174
A-n69-k9	1168	1159
A-n80-k10	1764	1763

BLIB		
Problem	Lit.	SfB
B-n50-k8	1313	1312
B-n51-k7	1104	1032
B-n57-k7	1278	1153
B-n63-k10	1537	1497
B-n64-k9	862	861
B-n66-k9	1374	1316
B-n67-k10	1033	1032
B-n68-k9	1304	1272
B-n78-k10	1266	1221

COUNTRYLIB		
Problem	Lit.	SfB
FRB-n51-k5	4197	4195
FRC-n51-k5	3774	3772

EILLIB		
Problem	Lit.	SfB
E-n30-k4	–	503
E-n76-k7	683	682
E-n76-k10	832	830
E-n76-k14	1032	1021
E-n101-k8	817	815
E-n101-k14	1077	1070

FISHERLIB		
Problem	Lit.	SfB
F-n45-k4	728	721
F-n72-k4	238	237
F-n135-k7	1165	1148

MLIB		
Problem	Lit.	SfB
M-n121-k7	1065	1034
M-n151-k12	1053	1021
M-n200-k16	–	1530
M-n200-k17	–	1303

Acknowledgment

I kindly acknowledge huge grants of computation time by the John von Neumann - Institute for Computing at the Forschungszentrum Jülich (former Höchstleistungsrechenzentrum) and the Swiss Federal Institute of Technology Zurich (ETHZ). Part of the work has been supported by the Swiss National Science Foundation.

References

[1] S. Kirkpatrick, C. D. Gelatt Jr. and M. P. Vecchi, Science **220**, 671 (1983).

[2] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, J. Chem. Phys. **21**, 1087 (1953).

[3] K. Binder and D. W. Heermann, *Monte Carlo Simulation in Statistical Physics*, 2nd edition, Springer (Berlin, Germany, 1992).

[4] G. Dueck and T. Scheuer, J. Comp. Phys. **90**, 161 (1990).

[5] J. Schneider, *Effiziente parallelisierbare physikalische Optimierungsverfahren*, PhD thesis (University of Regensburg, Germany, 1999).

[6] P. Moscato and J. F. Fontanari, Phys. Lett. A **146**, 204 (1990).

[7] E. Schöneburg, F. Heinzmann and S. Feddersen, *Genetische Algorithmen und Evolutionsstrategien*, Addison-Wesley (Bonn, Germany, 1994).

[8] J. Schneider, *Parallelisierung physikalischer Optimierungsverfahren*, diploma thesis (University of Regensburg, Germany, 1995).

[9] J. Schneider, Ch. Froschhammer, I. Morgenstern, Th. Husslein and J. M. Singer, Comp. Phys. Comm. **96**, 173 (1996).

[10] G. Reinelt, *The Traveling Salesman*, Lecture Notes in Computer Science 840, Springer (Berlin, Germany, 1994).

[11] P. F. Stadler and W. Schnabl, Phys. Lett. A **161**, 337 (1992).

[12] `http://www.informatik.uni-heidelberg.de/groups/comopt/software/TSPLIB95.`

[13] M. Grötschel and O. Holland, Math. Prog. **51**, 141 (1991).

[14] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman and L. Troyansky, Nature **400**, 133 (1999).

[15] `http://www-apache.imag.fr/~paugerat/VRP/INSTANCES.`

[16] J. Schneider, J. Britze, A. Ebersbach, I. Morgenstern and M. Puchta, Int. J. Mod. Phys. C **11**, 949 (2000).