λ-Opt Neural Approaches to Quadratic Assignment Problems

Shin Ishii
Nara Institute of Science and Technology, Ikoma-shi, Nara, 630-0101 Japan and ATR Human Information Processing Research Laboratories, Soraku-gun, Kyoto, 619-0288 Japan

Hirotaka Niitsuma
Nara Institute of Science and Technology, Ikoma-shi, Nara, 630-0101 Japan

In this article, we propose new analog neural approaches to combinatorial optimization problems, in particular, quadratic assignment problems (QAPs). Our proposed methods are based on an analog version of the λ-opt heuristics, which simultaneously changes assignments for λ elements in a permutation. Since we can take a relatively large λ value, our new methods can achieve a middle-range search over possible solutions, and this helps the system neglect shallow local minima and escape from local minima. In experiments, we have applied our methods to relatively large-scale (N = 80–150) QAPs. Results have shown that our new methods are comparable to the present champion algorithms; for two benchmark problems, they are obtain better solutions than the previous champion algorithms.

1 Introduction

We propose new analog neural approaches to combinatorial optimization problems. In particular, we deal with quadratic assignment problems (QAPs) (Burkard, Karisch, & Rendl, 1997), which are known to be very difficult combinatorial optimization problems. Our proposed approaches can also be applied to various combinatorial optimization problems where each solution is represented as a permutation. Examples are traveling salesman problems (TSPs).

In the case of a TSP, a solution is represented by aligning the city indices along the salesman’s route, which is a permutation of the set of city indices. A solution to a QAP can also be represented by a permutation. In conventional neural approaches, however, an Ising (binary) spin system (Hopfield & Tank, 1985; Bilbro et al., 1989) or a Potts spin system (Peterson & Söderberg, 1989; Ishii & Sato, 1997) has been used to represent a solution. The constraints for the permutation are then implemented as “soft” constraints; penalty terms for violations are added to the objective function. This implementation often produces infeasible solutions, one of the

reasons that these approaches are not very good when the problem size is expanded.

In some studies, on the other hand, all of the permutation constraints are treated as “hard” constraints; they are implemented to be automatically satisfied. Yuille and Kosowski (1994) proposed a gradient descent method in order to obtain a solution permutation for linear assignment problems. Rangarajan, Gold, and Mjolsness (1996) proposed a discrete-time algorithm called “soft-assign” in order to obtain a solution permutation for QAPs. We (Ishii & Sato, 1995, 1996) also proposed a similar algorithm called the doubly constrained network (DCN).1 Rangarajan et al. (1997) proved that their soft-assign, when combined with a deterministic annealing procedure, converges. In these studies, the space of possible solutions is almost equivalent to that of the permutations. The obtained solutions are therefore feasible in general, and they are better (Ishii & Sato, 1996, 1998; Gold & Rangarajan, 1996) than the solutions that would be produced by the Ising or Potts spin approach. Even in DCN (≈ soft-assign) combined with a deterministic annealing procedure, several problems exist. First, since the algorithm is deterministic, the system cannot escape from a local minimum by itself. In addition, when starting at a high temperature, there is no variety to the solution even if the initial conditions are variously prepared (Ishii & Sato, 1996).2 Second, in order to obtain a valid permutation, a careful deterministic annealing is needed, especially at low temperatures. Such features prevent these approaches from achieving further improvements in the solution.

In our new approach, we apply a replacement to a permutation as a basic operation. In addition, we introduce nonequilibrium dynamics to DCN. These modifications overcome the problems. The nonequilibrium dynamics allows the system to escape from local minima, and this results in variety to a solution. The basic replacement operation ensures that the system retrieves only valid permutations.

When the basic operation is an exchange of two elements in a permutation, the operation is called a 2-opt search. When λ (≥ 2) elements in a permutation are changed at once, the operation is called a λ-opt search (Lin & Kernighan, 1973; Martin, Otto, & Feltman, 1992). The 2-opt search is a very simple heuristic algorithm, but it easily falls into a local minimum. A neural network approach based on the 2-opt heuristics was proposed by

---

1 The DCN (≈ soft-assign) algorithm was first proposed by Gold, Rangarajan and Mjolsness (1994). Our articles (Ishii & Sato, 1995, 1996) are later, although they describe a convergence proof for a subprocedure, bifurcation discussion, and TSP (N ≤ 200) simulation results. Therefore, it is appropriate to call this algorithm soft-assign. Since this study is based on our DCN formulation (Ishii & Sato, 1996), however, the algorithm is sometimes called DCN in this article.

2 Starting at low temperature produces solution variety, depending on the initial conditions. However, this scheme does not produce good solutions. In this article, the deterministic annealing procedure is assumed to start at sufficiently high temperature.
Hasegawa, Ikeguchi, and Aihara (1998). Tabu list (Taillard, 1995) and chaos (Hasegawa et al., 1998) have been proposed for an escape from local minima. On the other hand, \( \lambda \)-opt (\( \lambda > 2 \)) heuristics get computationally heavy as \( \lambda \) becomes large.

Our proposed methods are based on an analog version of the \( \lambda \)-opt heuristics. Since we employ analog neural approaches based on the DCN formulation, it is possible to employ a fairly large \( \lambda \) (e.g., about \( N/2 \) where \( N \) is the problem’s size). The fairly large \( \lambda \) value enables the system to search for good permutations over a middle-range region and prevents the system from falling into shallow local minima around the present system state. In addition, our methods naturally introduce nonequilibrium dynamics, which enables the system to search for various solutions one after another in the possible permutation space.

In experiments, we apply our new methods to relatively large-scale (\( N = 80–150 \)) QAPs taken from QAPLIB (Burkard et al., 1997), which is a standard set of QAP benchmark problems. Our methods are comparable to the present champion algorithms, which are based on tabu search (Taillard, 1991; Battiti & Tecchiolli, 1994), genetic algorithms (Fleurent & Ferland, 1994), and simulated annealing (Amin, 1999). Moreover, for two benchmark problems, our methods are able to better the champion algorithms. Consequently, our methods can be considered as some of the stronger algorithms for QAPs.

\section*{2 \( \lambda \)-DCN}

\subsection*{2.1 Quadratic Assignment Problem}
QAPs are known to be very difficult combinatorial optimization problems (Burkard et al., 1997). Naturally, they belong to the class of NP-hard problems. A typical instance of QAPs is a facility location problem, in which a set of facilities is assigned to an equal number of locations at the minimum cost. For each pair of facilities, an amount of flow is given, and for each pair of locations, a distance is given. The cost is defined as the summation of the product of the flow between two facilities and the distance between the locations to which the facilities are assigned. Many combinatorial optimization problems, such as TSPs, maximum clique problems, and graph isomorphism problems, are special and easy QAP examples.

Let \( D \) and \( C \) denote \( N \)-by-\( N \) distance and flow matrices, respectively. Here, a QAP with a problem size \( N \) is defined by

\[
\min_{p \in \Pi} \sum_{a=1}^{N} \sum_{b=1}^{N} D_{a,b} C_{p(a), p(b)},
\]  \hspace{1cm} (2.1)

where \( \Pi \) is the set of all permutations of \( \{1, 2, \ldots, N\} \), and \( p(a) \) gives the element assigned to the location \( a \) in a permutation \( p \in \Pi \).
To deal with problem 2.1, we define an \( N \times N \) assignment matrix:

\[
S_{a,n} = \begin{cases} 1 & \text{if } p(a) = n \text{ and } S_{a,n} = 0 \text{ otherwise.} \\
0 & \text{otherwise.}
\end{cases}
\]

The constraints—\( \sum_{a=1}^{N} S_{a,n} = 1 \ (a = 1, \ldots, N) \) and \( \sum_{n=1}^{N} S_{a,n} = 1 \ (n = 1, \ldots, N) \)—should be satisfied for the assignment matrix \( S \) to represent a valid permutation \( p(a) \). By using the assignment matrix \( S \), we can define problem 2.1 as a minimization of an objective energy function,

\[
E_{\text{obj}}(S) = \frac{1}{2} \sum_{a,b,n,m=1}^{N} D_{a,b} C_{n,m} S_{a,n} S_{b,m},
\]

which is subject to the assignment matrix constraints. If the facility matrix is of the form \( F_{n,m} = (\delta_{n,(m+1)} + \delta_{n,(m-1)})/2 \), objective energy function 2.2 is equivalent to that of an \( N \)-city TSP. Here, \( \delta_{i,j} \) is Kronecker’s delta. Namely, a TSP is a special QAP.

We define a modified energy function,

\[
E(S) = E_{\text{obj}}(S) + \frac{\eta}{2} \sum_{a,n=1}^{N} S_{a,n}(1 - S_{a,n}),
\]

where \( \eta \) is a positive constant. Although the second term is always zero and hence meaningless for a binary \( S \), studies have shown (Hopfield & Tank, 1985; Peterson & Söderberg, 1989; Ishii & Sato, 1996; Rangarajan et al., 1996) that this quadratic term helps stabilize the analog algorithm shown later.

Next, we define an \( N \times N \) permutation matrix by

\[
Y_{a,b} = \begin{cases} 1 & \text{if element } p(b) \text{ changes its location from } b \text{ to } a \\
0 & \text{otherwise.}
\end{cases}
\]

Although an assignment matrix \( S \) can be regarded as a permutation matrix applied to the identity assignment, \( p(n) = n \ (n = 1, \ldots, N) \), we distinguish assignment matrices and permutation matrices for a convenient description. For the permutation matrix \( Y \), there are also constraints:

\[
\sum_{a=1}^{N} Y_{a,b} = 1 \quad (b = 1, \ldots, N) \quad (2.5a)
\]

\[
\sum_{b=1}^{N} Y_{a,b} = 1 \quad (a = 1, \ldots, N). \quad (2.5b)
\]

In addition, we consider a constraint,

\[
\sum_{a=1}^{N} Y_{a,a} = N - \lambda, 
\]

\[
(2.6)
\]
where \(0 \leq \lambda \leq N\) is a constant integer. Constraint 2.6 restricts the number of elements whose location is changed by the permutation \(Y\) to \(\lambda\). In the following, a permutation matrix \(Y\) subject to constraints 2.5 and 2.6 is called a \(\lambda\)-permutation matrix. When a permutation matrix \(Y\) applies to an assignment matrix \(S\), the new assignment matrix \(S'\) is given by

\[
S' = YS. \tag{2.7}
\]

Next, we consider a set of \(\lambda\)-permutation matrices: \(\Lambda = \{ (Y^k, p^k) \mid k = 1, \ldots\}\), where \(p^k\) is the probability for the permutation \(Y^k\). The ensemble average for the new assignment matrices applied by this set of permutation matrices is given by

\[
\langle S' \rangle_{\Lambda} = \langle Y \rangle_{\Lambda} S, \tag{2.8}
\]

where \(\langle \cdot \rangle_{\Lambda}\) is the mean with respect to the permutation set: \(\langle f(k) \rangle_{\Lambda} = \sum_k f(k)p^k\). We use the notations: \(V \equiv \langle S' \rangle_{\Lambda}\) and \(X \equiv \langle Y \rangle_{\Lambda}\).

Let us now consider a minimization of the energy function, \(E(S')\), with respect to the new assignments. To make this discrete problem a continuous one, we introduce a free energy function:

\[
F \equiv \langle E(S') \rangle_{\Lambda} - TH \tag{2.9a}
\]

\[
H \equiv -\sum_{S} p(S') \log p(S') \approx -\sum_{a,n=1}^{N} V_{a,n} \log V_{a,n}, \tag{2.9b}
\]

where \(p(\cdot)\) denotes the probability for that configuration. \(T\) is the temperature and \(H\) is the entropy of the possible configurations.

For the time being, we deal with a more general problem, where the energy function has a quadratic form,

\[
E(S') = \frac{1}{2} \sum_{a,n,b,m=1}^{N} W_{a,n;b,m} S'_{a,n} S'_{b,m} + \sum_{a,n=1}^{N} I_{a,n} S'_{a,n}, \tag{2.10}
\]

where the weight matrix is symmetric, that is, \(W_{a,n;b,m} = W_{b,m;a,n} (a, n, b, m = 1, \ldots, N)\). A QAP whose energy function is given by 2.3 is one such problem. For the energy function, 2.10, the following approximation can be performed:

\[
\langle E(S') \rangle_{\Lambda} \approx \frac{1}{2} \sum_{a,n,b,m=1}^{N} W_{a,n;b,m} V_{a,n} V_{b,m} + \sum_{a,n=1}^{N} I_{a,n} V_{a,n}
\]

\[
= \frac{1}{2} \sum_{a,b,c,d=1}^{N} \left( \sum_{n,m=1}^{N} W_{a,n;b,m} S_{c,n} S_{d,m} \right) X_{a,c} X_{b,d}
\]
where $\tilde{W}_{a,b,c,d}(S) = \sum_{n,m} W_{a,n,c,m} S_{b,n} S_{d,m}$ and $\tilde{I}_{a,b}(S) = \sum_{n} I_{a,n} S_{b,n}$. The new weight matrix is also symmetric: $\tilde{W}_{a,b,c,d}(S) = \tilde{W}_{c,d,a,b}$. In equations 2.9b and 2.11, we use a similar approximation to the mean-field theory (Bilbro et al., 1989; Peterson & Söderberg, 1989; Yuille & Kosowski, 1994), which is not rigorously applicable to the configuration space of the assignment matrices.

The entropy is also approximated as

$$H \approx - \sum_{a,n} V_{a,n} \log V_{a,n} = - \sum_{a,n} X_{a,p^{-1}(a)} \log X_{a,p^{-1}(a)}$$

$$= - \sum_{a \neq b} X_{a,b} \log X_{a,b} \equiv \tilde{H}(X),$$

where $p$ is the permutation represented by $S$. Similarly, constraints 2.5 and 2.6 are also represented by the average permutation matrix $X$.

Accordingly, the new continuous problem is defined by

minimize $\tilde{F}(X; S) \equiv \tilde{E}(X; S) - T\tilde{H}(X)$  \hspace{1cm} (2.13a)

subject to

$\sum_{a=1}^{N} X_{a,b} = 1 \hspace{1cm} (b = 1, \ldots, N)$  \hspace{1cm} (2.13b)

$\sum_{b=1}^{N} X_{a,b} = 1 \hspace{1cm} (a = 1, \ldots, N)$  \hspace{1cm} (2.13c)

$\sum_{a=1}^{N} X_{a,a} = N - \lambda$  \hspace{1cm} (2.13d)

The new problem is a minimization of the free energy function, 2.13a, which is defined by the average permutation and not by the average assignment. In addition, it is very important that the problem is not a global one. It is a local problem that depends on the previous assignment $S$. In problem 2.13, the entropy, 2.12, with constraints 2.13b and 2.13c prevents the average permutation $X$ from going out of its domain $[0, 1]^N$. In this sense, it functions as a barrier function (Luenberger, 1989; Yuille & Kosowski, 1994). The constant $\lambda$ is not necessarily an integer in this continuous problem.
With the Lagrange method, a solution to problem 2.13 can be given by the following simultaneous equations:

\[
X_{a,b} = \frac{U_{a,b}}{\alpha_a \beta_b \gamma_{a,b}} \quad (2.14a)
\]

\[
U_{a,b} \equiv \exp \left( -\frac{1}{T} \frac{\partial \hat{E}(X; S)}{\partial X_{a,b}} \right) \quad (2.14b)
\]

\[
\alpha_a = \sum_b \frac{U_{a,b}}{\beta_b \gamma_{a,b}} \quad (2.14c)
\]

\[
\beta_b = \sum_a \frac{U_{a,b}}{\alpha_a \gamma_{a,b}} \quad (2.14d)
\]

\[
\gamma = \frac{1}{N - \lambda} \sum_a \frac{U_{a,a}}{\alpha_a \beta_a} \quad (2.14e)
\]

where \( \alpha_a (a = 1, \ldots, N) \), \( \beta_b (b = 1, \ldots, N) \) and \( \gamma \) are Lagrange multipliers. For the energy function, 2.11, \( \partial \hat{E}/\partial X_{a,b} = \sum_{c,d} \tilde{W}_{a,b,c,d} X_{c,d} + \tilde{I}_{a,b} \). The equations defined by 2.14 are called \( \lambda \)-DCN equations.

2.3 Basic Algorithm. The following basic algorithm works well in a practical sense for solutions to \( \lambda \)-DCN equations 2.14.

1. Set \( t \) at 0. Set \( X(0) \) to be

\[
X_{a,b}(0) = \begin{cases} 
(1 - \lambda/N)(1 + \epsilon_{a,b}) & \text{if } a = b \\
\lambda(1 + \epsilon_{a,b})/(N(N - 1)) & \text{if } a \neq b,
\end{cases} \quad (2.15)
\]

where \( \epsilon_{a,b} \) denotes small noise (e.g., a uniform random value in \([-0.1, 0.1])\).

2. For all \( a, b \), calculate:

\[
U_{a,b} = \exp \left( -\frac{1}{T} \frac{\partial \hat{E}(X(t); S)}{\partial X_{a,b}(t)} \right). \quad (2.16)
\]

3. Set \( \beta_{b}^{\text{old}} \) to be \( \sum_a \frac{U_{a,b}}{\alpha_a(t) \gamma(t)_{a,b}} \) for all \( b \), and set \( \gamma^{\text{old}} \) to be \( \frac{1}{N - \lambda} \sum_a \frac{U_{a,a}}{\alpha_a(t) \beta_a(t)} \).

4. The following substeps are iterated:

a. For all \( a \), calculate:

\[
\alpha_a^{\text{new}} = \sum_{b=1}^{N} \frac{U_{a,b}}{\beta_b^{\text{old}}(\gamma^{\text{old}})_{a,b}} \quad (2.17)
\]
b. For all \( b \), calculate:

\[
\beta_{b}^{\text{new}} = \sum_{a=1}^{N} \frac{U_{a,b}}{\alpha_{a}^{\text{new}} (\gamma_{\text{old}})^{b_{a,b}}}.
\]  

(2.18)

c. Calculate:

\[
\gamma^{\text{new}} = \frac{1}{N - \lambda} \sum_{a=1}^{N} \frac{U_{a,a}}{\alpha_{a}^{\text{new}} \beta_{a}^{\text{new}}}.
\]  

(2.19)

If each of \( \alpha_{a} \), \( \beta_{b} \), and \( \gamma \) converges, set \( \alpha_{a}(t+1) \), \( \beta_{b}(t+1) \), and \( \gamma(t+1) \) to those values. Then rescale \( \alpha(t+1) \) so that \( \sum_{a} \alpha_{a}(t+1) = 1 \).

5. For all \( a, b \), calculate:

\[
X_{a,b}(t+1) = \frac{U_{a,b}}{\alpha_{a}(t+1) \beta_{b}(t+1) \gamma(t+1)^{b_{a,b}}}.
\]  

(2.20)

6. If \( X(t) \) converges or \( t \) is equal to \( t_{\text{max}} \), exit the algorithm. Otherwise, add 1 to \( t \) and go to step 2.

This algorithm does not necessarily converge. The average permutation matrix \( X \) often exhibits a quasi-periodic or periodic oscillation. Therefore, the number of iterations is limited to \( t_{\text{max}} \), which is set at 100 in the experiments in section 4. When constraint 2.13d (and hence step 4c) is removed, the above algorithm is almost equivalent to the soft-assign (\( \approx \) DCN). In this case, there are convergence proofs for step 4 (Sinkhorn, 1964),\(^3\) and for the whole basic algorithm (Rangarajan et al., 1997). With step 4c, however, the convergence of step 4 has not been proved yet. Nevertheless, our computer simulations imply that step 4, including step 4c, always converges. The effect of step 4c will be experimentally investigated in section 4.

2.4 \( \lambda \)-DCN Algorithm. When the temperature \( T \) is small, the average permutation matrix \( X \) obtained by the above basic algorithm tends to be close to a vertex of the domain hypercube. Then the matrix is regarded as signifying one of the valid \( \lambda \)-permutation matrices. Moreover, we often obtain several valid permutation matrices during a single process of the basic algorithm. Among these, we choose the best permutation matrix: the one that minimizes the objective energy function (equation 2.2 for a QAP). This process retrieves a good new assignment matrix that is an application of one of the \( \lambda \)-permutation matrices to the previous assignment matrix \( S \). However, the average permutation matrix \( X \) sometimes fails to approach

\(^3\) Our previous paper (Ishii & Sato, 1996) also provided another convergence proof based on the Lyapunov method.
any vertex of the hypercube. A typical case is that $X$ falls into a two-cycle oscillation, and neither state signifies a valid permutation. In such a case, the result of the basic algorithm is regarded as a failure. When a new assignment is obtained, the assignment is improved by the 2-opt heuristics. This additional procedure needs negligible computation time.

Our proposed procedure, which is called $\lambda$-DCN, repeats the process. Figure 1 depicts this procedure. After a single process, a new assignment $S_2$, which is transformed from the previous assignment $S_1$ by a $\lambda$-permutation $Y_1$, is retrieved. The new assignment $S_2$ is expected to be a good one among the assignments that are reachable from $S_1$ by a $\lambda$-permutation. In the subsequent process, another assignment $S_3$ is retrieved from $S_2$ by calculating a good $\lambda$-permutation $Y_2$. When the basic algorithm fails to obtain $Y_2$, the starting assignment in the next process is again $S_2$. Due to the variation of the initial condition (step 1) in the basic algorithm, the next application of the basic algorithm will obtain a valid $\lambda$-permutation.

When the parameter $\lambda$ is equal to 2, this procedure corresponds to an analog version of the 2-opt heuristics, a local search algorithm for the nearest neighbors. However, by taking a relatively large $\lambda$ value, our $\lambda$-DCN algorithm can achieve a middle-range search for the possible assignments. This helps the system neglect shallow local minima around the previous assignment and enables a search of good but far-located local minima of the energy function. Furthermore, this procedure never terminates; nonequilibrium dynamics is introduced. The system is not trapped by any local minimum. In addition, the nonequilibrium dynamics naturally introduces variety to the solution. Therefore, as the number of iterations—applications of the basic algorithm—becomes large, improvements in the solution can be expected. We discuss this further in section 4.
3 $\lambda$-Interior DCN

Since the $\lambda$-DCN algorithm retrieves assignments using $\lambda$-permutations, it often ignores good local minima located in the nearby neighborhood around the current assignment.

To deal with this problem, we propose another method here, which is inspired by the interior point method (Karmarker, 1984). In this method, the equality constraint, 2.6, is replaced by the following inequality constraint,

$$\sum_{a=1}^{N} Y_{a,a} > N - \lambda,$$

which restricts the number of elements relocated by the permutation $Y$ to be less than or equal to $\lambda$. In order to implement this inequality constraint, the following barrier function is introduced into the free energy function, 2.13a:

$$\tilde{K}(X) = T \left( \sum_{a=1}^{N} X_{a,a} - M \right) \log \left( \sum_{a=1}^{N} X_{a,a} - M \right) + T(\theta - 1) \left( \sum_{a=1}^{N} X_{a,a} - M \right),$$

where $M = N - \lambda$ and $\theta (= 0 \sim 1)$ is a constant parameter. In this case, constraint 2.13d is removed.

Using barrier function 3.2, equations 2.14b and 2.14e in the $\lambda$-DCN equations are replaced by

$$U_{a,b} = \exp \left( -\frac{1}{T} \frac{\partial \tilde{E}(X; S)}{\partial X_{a,b}} - \theta \delta_{a,b} \right),$$

$$\gamma = \sqrt{M^2 + 4 \sum_{a} \frac{U_{a,a}}{\rho_a^2} - M}.$$

This derivation is described in the appendix. Then equation 2.16 in the basic algorithm in section 2.3 is replaced by

$$U_{a,b} = \exp \left( -\frac{1}{T} \frac{\partial \tilde{E}(X(t); S)}{\partial X_{a,b}(t)} - \theta \delta_{a,b} \right);$$

equation 2.19 is replaced by

$$\gamma_{\text{new}} = \sqrt{M^2 + 4 \sum_{a} \frac{U_{a,a(t)}}{\rho_a(t)^2} - M}.$$
Table 1: Solutions.

<table>
<thead>
<tr>
<th>Name</th>
<th>N</th>
<th>Champion</th>
<th>DCA</th>
<th>2-Opt</th>
<th>λ-DCN</th>
<th>λ-Inter DCN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tai80a</td>
<td>80</td>
<td>13557864</td>
<td>1.2</td>
<td>3.1</td>
<td>-0.060 (2568)</td>
<td>-0.040 (2481)</td>
</tr>
<tr>
<td>Tai100a</td>
<td>100</td>
<td>21123514</td>
<td>0.50</td>
<td>3.2</td>
<td>-0.089 (469)</td>
<td>-0.082 (5656)</td>
</tr>
<tr>
<td>Wil100</td>
<td>100</td>
<td>273038</td>
<td>0.27</td>
<td>0.73</td>
<td>0.46 (2109)</td>
<td>0.070 (2563)</td>
</tr>
<tr>
<td>Tho150</td>
<td>150</td>
<td>8133484</td>
<td>0.33</td>
<td>1.5</td>
<td>0.24 (7139)</td>
<td>0.28 (6956)</td>
</tr>
</tbody>
</table>

Notes: “Name” represents the problem’s name, “N” is the problem size, and “champion” is the previous best (champion) solution. “DCA” is the solution obtained by DCN annealing, “2-opt” is the best solution obtained by many 2-opt runs for various initial conditions, “λ-DCN” is the solution obtained by our λ-DCN method, and “λ-inter DCN” is the solution obtained by our λ-interior DCN method. Each solution is represented by the deviation (in percentage) from the previous champion solution. The parentheses denote number of basic algorithms for the best solution.

The other parts are the same as in the basic algorithm.

The modified basic algorithm searches for a good l-permutation where 0 ≤ l ≤ λ from the previous assignment S. Therefore, the new procedure simultaneously achieves a local search and a middle-range search in the vicinity of the previous assignment. The modified procedure is called the λ-interior DCN algorithm.

4 Experiments

Our proposed methods can be applied to any quadratic problem whose solutions are represented by permutations. In the case of a QAP whose energy function is given by equation 2.3, we set W_{a,n,b,m} = D_{a,b}C_{n,m} = \eta (a, n, b, m = 1, \ldots, N) and \lambda_{a,n} = \eta/2 (a, n = 1, \ldots, N). \eta is a small positive constant. Although \lambda is experimentally set at a good value, our experiential prescription is to set it at about N/2.

The benchmark problems used in the experiments are Tai80a (N = 80), Tai100a (N = 100), Wil100 (N = 100), and Tho150 (N = 150), all of which were taken from QAPLIB (Burkard et al., 1997). The distance and facility matrices for these problems are symmetric and dense. Tho150 is the largest problem with its dense matrices in QAPLIB. For Tai80a and Tai100a, the champion algorithms (Taillard, 1991; Battiti & Tecchiolli, 1994) are based on the tabu search. For Wil100, the champion algorithm (Fleurent & Ferland, 1994) is a combined algorithm of a genetic algorithm and the tabu search. For Tho150, the champion algorithm (Amin, 1998) is called simulated jumping. The QAPLIB web site maintains the champion data for these problems.

Table 1 shows the experimental results. The value in each column is a deviation in percentage from the present champion solution. For comparison, the table also contains solutions obtained by our previous approach (Ishii & Sato, 1998), DCN annealing, and the best solutions that the 2-opt heuristics could obtain from many runs. Note that DCN annealing is almost
equivalent to the algorithm proposed by Rangarajan et al. (1996), and its solutions are much better than those produced by the Potts spin network (i.e., soft-max) combined with a deterministic annealing procedure (Peterson & Söderberg, 1989). With DCN annealing, λ-DCN, and λ-interior DCN, the 2-opt heuristics improved the solutions. The table also shows the applications of the basic algorithm until the best solutions are retrieved for λ-DCN and λ-interior DCN.

In Table 1, the results of our new methods are comparable to the champion solutions. For Tai80a and Tai100a, both of the proposed methods achieved better solutions than the previous champion algorithms.

Our methods seem faster than, or at least as fast as, the previous champion algorithms, although a rigorous comparison is difficult due to the lack of description of the computation time in the literature and the different computer environments. For Tai100a, λ-DCN took 45 minutes to obtain its best solution on a Silicon Graphics Origin 2000 computer. This computation time was for 469 applications of the basic algorithm. Ten thousand applications of the basic algorithm required about 12 hours and 20 hours in λ-DCN and λ-interior DCN, respectively. For the same problem, DCN annealing took 22 minutes for its best solution shown in Table 1. The soft-max combined with deterministic annealing took 32 minutes for its best solution that was 1.03% worse than the champion solution. Our new methods are slower than the DCN and soft-max approaches. Since DCN annealing is almost deterministic, there is no variety to a solution even if the initial conditions are variously prepared (Ishii & Sato, 1996). On the other hand, our new methods naturally introduce nonequilibrium dynamics that search for possible solutions by solving local problems one after another.

The advantage of our methods is not only an automatic multiple restart scheme. Figure 2 compares solutions obtained by the λ-DCN algorithm (solid lines) and the best solutions produced by the multiple restarts of the λ-DCN basic algorithm from random initial assignments (dashed lines). In the multiple restarts, the number of initial assignments is set to be equal to the number of assignments obtained by λ-DCN. Figure 2 implies that the multiple restarts cannot achieve significant improvement even when

---

4 For Tai100a, the champion algorithm (Battiti & Tecchiolli, 1994) took about 10 hours on a Silicon Graphics Iris computer for a solution that was 0.17% worse than the present best solution. To obtain the present best solution, the algorithm would need more computation time. The same literature mentioned that the champion algorithm for Tai80a (Taillard, 1991) was a little slower. The champion algorithm for Wil100 (Fleurent & Ferland, 1994) seems to have taken about three days to obtain a solution for Tai100a using a SUN SPARC 10 computer.

5 In these experiments, DCN annealing and soft-max annealing were applied 137 and 162 times, respectively, of the subprocedure, which is similar to the λ-DCN basic algorithm.

6 A more detailed algorithm is to (1) randomly prepare an assignment, (2) improve the assignment by the 2-opt heuristics, and (3) apply the λ-DCN basic algorithm only once. Repeat this procedure.
Figure 2: Comparison between $\lambda$-DCN and multiple restarts of basic algorithm. Abscissa and ordinate denote the applications of basic algorithm and best solution until that number, respectively. Solid lines are $\lambda$-DCN algorithm, and dashed lines are best solutions by multiple restarts of the basic algorithm from randomly prepared initial assignments. For each number of applications, the number of initial assignments used by multiple restarts is set to be equal to the number of valid assignments obtained by $\lambda$-DCN. Namely, the number of candidate assignments is even in two methods. Six experiments conducted for each method (a line).

The applications increase, while our $\lambda$-DCN can. The improvements due to sequential applications of the basic algorithm are very important. Table 2 shows the average and the standard deviation of solutions obtained by $\lambda$-DCN for various applications of the basic algorithm.

Finally, let us briefly mention the effect of the constraint, equation 2.13d or 3.2, for the diagonal sum of the average permutation matrix $X$. In the experiments for $\lambda$-DCN and $\lambda$-interior DCN, step 4 in the basic algorithm always converges, although the convergence is very slow in some cases.\footnote{In about 0.1% cases in both methods, the convergence needed more than 1,000 iterations.} Therefore, the basic algorithm is practically valid. If the convergence criterion for step 4 in $\lambda$-DCN or $\lambda$-interior DCN is set as tight as that in DCN, the number of iterations needed in that step becomes larger than that in DCN.
Table 2: Solutions Against Number of Iterations.

<table>
<thead>
<tr>
<th>Number of iterations</th>
<th>0</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution (%)</td>
<td>5.12 ± 0.26</td>
<td>0.47 ± 0.09</td>
<td>0.28 ± 0.04</td>
<td>0.24 ± 0.08</td>
<td>0.19 ± 0.08</td>
<td>0.11 ± 0.11</td>
</tr>
</tbody>
</table>

Notes: This table shows the average and the standard deviation of the Tai100a solutions obtained by the $\lambda$-DCN algorithm for various application numbers of the basic algorithm. Ten random initial assignments were prepared. Solution statistics are the difference (in percentage) from the previous champion solution. When the number of iterations is 0, the solutions are the initial random assignments improved by the 2-opt heuristics.

due to the effect of step 4c. However, $\lambda$-DCN and $\lambda$-interior DCN do not need a very tight convergence criterion for step 4, because their dynamics, which is based on permutations, do not need a very accurate calculation. On the other hand, the solutions obtained by DCN annealing are highly dependent on the convergence criterion. In the experiments for Tai100a, the average (standard deviation) of iteration numbers for step 4 were 13.1 ($\pm$ 30.1), 3.4 ($\pm$ 15.6), and 23.2 ($\pm$ 35.1), in DCN annealing, $\lambda$-DCN, and $\lambda$-interior DCN, respectively. In the same experiments, the basic algorithm converged in 76% and 48% in $\lambda$-DCN and $\lambda$-interior DCN, respectively.

5 Conclusion

When $\lambda$ is relatively large, it is computationally difficult to search for a good assignment over all of the assignments reachable by $\lambda$-permutations from the current assignment. We proposed a couple of neural methods that partially achieve this mechanism, by considering the free energy function dependent on the current assignment. These methods are $\lambda$-DCN and $\lambda$-interior DCN. The new methods are also improvements of the previous DCN ($\approx$ soft-assign) annealing in two ways. First, since nonequilibrium dynamics is introduced, the system is not trapped by any local minimum, and the system can retrieve various candidates. Moreover, the solution is expected to be improved by sequential applications of the $\lambda$-permutations. Second, since we take a permutation as a basic operation, the retrieved solutions always represent valid permutations. This is one of the reasons that our methods work well.

In the experiments, we applied our methods to relatively large-scale ($N = 80–150$) QAP benchmark problems. Consequently, the new methods were found to be comparable to the present champion algorithms; for two benchmark problems, they were better than the champion algorithms. We therefore conclude that our new methods are some of the stronger algorithms for QAPs.
Appendix

Here, we derive the \(\lambda\)-interior DCN equations defined by equations 2.14a, 3.3b, 2.14c, 2.16d, and 3.3e. To minimize \(\hat{F}(X; S) = \hat{E}(X; S) - T\hat{H}(X) + \hat{K}(X)\) subject to constraints 2.13b and 2.13c, we define a Lagrange function:

\[
L = \hat{\mathcal{F}} + \sum_{a=1}^{N} A_a \left( \sum_{b=1}^{N} X_{a,b} - 1 \right) + \sum_{b=1}^{N} B_b \left( \sum_{a=1}^{N} X_{a,b} - 1 \right),
\]

(A.1)

where \(A_a (a = 1, \ldots, N)\) and \(B_b (b = 1, \ldots, N)\) are Lagrange multipliers. A stationary condition of the Lagrange function is given by

\[
\frac{\partial L}{\partial X_{a,b}} = \frac{\partial \hat{E}}{\partial X_{a,b}} + T \left[ \log X_{a,b} + 1 + \delta_{a,b} \left( \log \left( \sum_{a} X_{a,a} - M \right) + \theta \right) \right]
\]

\[+ A_a + B_b = 0, \quad (A.2)\]

with constraints 2.13b and 2.13c. Equation A.2 is solved as

\[
X_{a,b} = \exp \left( -\frac{1}{T} \frac{\partial \hat{E}}{\partial X_{a,b}} \right) / (\alpha_a \beta_b) \quad \text{if } a \neq b \quad (A.3a)
\]

\[
X_{a,a} \left( \sum_{c} X_{c,c} - M \right) = \exp \left( -\frac{1}{T} \frac{\partial \hat{E}}{\partial X_{a,a}} - \theta \right) / (\alpha_a \beta_a). \quad (A.3b)
\]

where \(\alpha_a \equiv \exp(A_a / T + 1)\) and \(\beta_b \equiv \exp(B_b / T)\) are the converted Lagrange multipliers. Equation A.3a is equivalent to 2.14a and 3.3b for \(a \neq b\). The summation of equation A.3b over \(a\) gives a quadratic equation for \(\sum_a X_{a,a}\). A solution to the quadratic equation is given by

\[
\sum_a X_{a,a} = \frac{M + \sqrt{M^2 + 4 \sum_a \frac{U_{a,a}}{\alpha_a \beta_a}}}{2}. \quad (A.4)
\]

By defining \(\gamma = \sum_a X_{a,a} - M\), equation 3.3e is obtained. Equation 3.3b for \(a = b\) is determined from equation A.3b and the definition of \(\gamma\).

Acknowledgments

We thank the referees and the editor for their insightful comments in improving the quality of this article. We are grateful to Masa-aki Sato for valuable discussions.

References


λ-Opt Neural Approaches


Received November 16, 1998; accepted September 23, 1999.