Communicated by Eric Mjolsness

Airline Crew Scheduling with Potts Neurons

Martin Lagerholm
Carsten Peterson
Bo Söderberg
Department of Theoretical Physics, University of Lund, Sölvegatan 14A, S-223 62 Lund, Sweden

A Potts feedback neural network approach for finding good solutions to resource allocation problems with a nonfixed topology is presented. As a target application, the airline crew scheduling problem is chosen. The topological complication is handled by means of a propagator defined in terms of Potts neurons. The approach is tested on artificial random problems tuned to resemble real-world conditions. Very good results are obtained for a variety of problem sizes. The computer time demand for the approach only grows like (number of flights)^3. A realistic problem typically is solved within minutes, partly due to a prior reduction of the problem size, based on an analysis of the local arrival and departure structure at the single airports.

1 Introduction

In the past decade, feedback neural networks have emerged as a useful method to obtain good approximate solutions to various resource allocation problems (Hopfield & Tank, 1985; Peterson & Söderberg, 1989; Gislén, Söderberg, & Peterson, 1989, 1992). Most applications have concerned fairly academic problems like the traveling salesman problem (TSP) and various graph partition problems (Hopfield & Tank, 1985; Peterson & Söderberg, 1989). Gislén, Söderberg, and Peterson (1989, 1992) explored high school scheduling. The typical approach proceeds in two steps: (1) map the problem onto a neural network (spin) system with a problem-specific energy function, and (2) minimize the energy by means of deterministic mean field (MF) equations, which allow for a probabilistic interpretation. Two basic mapping variants are common: a hybrid (template) approach (Durbin & Willshaw, 1987) and a "purely" neural one. The template approach is advantageous for low-dimensional geometrical problems like the TSP, whereas for generic resource allocation problems, a purely neural Potts encoding is preferable.

A challenging resource allocation problem is airline crew scheduling, where a given flight schedule is to be covered by a set of crew *rotations*, each consisting of a connected sequence of flights (*legs*), starting and ending at a

Neural Computation 9, 1589–1599 (1997)

given home base (*hub*). The total crew waiting time is then to be minimized, subject to a number of restrictions on the rotations. This application differs strongly from high school scheduling (Gislén, Söderberg, & Peterson, 1989, 1992) in the existence of nontrivial topological restrictions. A similar structure occurs in multitask telephone routing.

A common approach to this problem is to convert it into a set covering problem, by generating a large pool of legal rotation templates and seeking a subset of the templates that precisely covers the entire flight schedule. Solutions to the set covering problem are then found with linear programming techniques or feedback neural network methods (Ohlsson, Peterson, & Söderberg, 1993). A disadvantage with this method is that the rotation generation for computational reasons has to be nonexhaustive for a large problem; thus, only a fraction of the solution space is available.

The approach to the crew scheduling problem developed in this article is quite different and proceeds in two steps. First, the full solution space is narrowed down using a reduction technique that removes a large part of the suboptimal solutions. Then, a mean field annealing approach based on Potts neurons is applied; a novel key ingredient is the use of a propagator formalism for handling topology, leg counting, and so forth.

The method, which is explored on random artificial problems resembling real-world situations, performs well with respect to quality, with a computational requirement that grows like N_f^3, where N_f is the number of flights.

2 Nature of the Problem

Typically, a real-world flight schedule has a basic period of one week. Given such a schedule in terms of a set of N_f weekly flights, with specified times and airports of departure and arrival, a crew is to be assigned to each flight such that the total crew waiting time is minimized, subject to the constraints:

- Each crew must follow a connected path (a *rotation*) starting and ending at the hub (see Figure 1).
- The number of flight legs in a rotation must not exceed a given upper bound $L_{\max}$.
- The total duration (flight plus waiting time) of a rotation is similarly bounded by $T_{\max}$.

These are the crucial and difficult constraints. In a real-world problem, there are often some twenty additional ones, which we neglect for simplicity; they constitute no additional challenge from an algorithmic point of view.

Without the above constraints, the problem would reduce to that of minimizing waiting times independently at each airport (the *local problems*). These can be solved exactly in polynomial time, for example, by pairwise

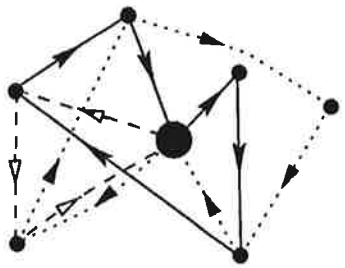

Figure 1: Schematic view of the three crew rotations starting and ending in a hub.

connecting arrival flights with subsequent departure flights. It is the global structural requirements that make the crew scheduling problem a challenge.

3 Reduction of Problem Size: Airport Fragmentation

Prior to developing our artificial neural network method, we will describe a technique to reduce the size of problem, based on the local flight structure at each airport.

With the waiting time between an arriving flight i and a departing flight j defined as

$$t_{ij}^{(w)} = \left(t_j^{(dep)} - t_i^{(arr)}\right) \bmod \text{period}, \tag{3.1}$$

the total waiting time for a given problem can change only by an integer times the period. By demanding a minimal waiting time, the local problem at an airport typically can be split up into independent subproblems, where each contains a subset of the arrivals and an equally large subset of the departures. For example, with A/D denoting arrival/departure, the time ordering $[AADDAD]$ can, if minimal waiting time is demanded, be divided into two subproblems: $[AADD]$ and $[AD]$. Some of these are trivial ($[AD]$), forcing the crew of an arrival to continue to a particular departure. The minimal total wait time for the local problems is straightforward to compute and will be denoted by $T_{\min}^{\text{wait}}$.

Similarly, by demanding a solution (assuming one exists) with $T_{\min}^{\text{wait}}$ also for the constrained global problem, this can be reduced as follows:

- *Airport fragmentation.* Divide each airport into *effective airports* corresponding to the nontrivial local subproblems.
- *Flight clustering.* Join every forced sequence of flights into one effective

composite flight, which will thus represent more than one leg and have a formal duration defined as the sum of the durations of its legs and the waiting times between them.

The reduced problem thus obtained differs from the original problem only in an essential reduction of the suboptimal part of the solution space; the part with minimal waiting time is unaffected by the reduction. The resulting information gain, taken as the natural logarithm of the decrease in the number of possible configurations, empirically seems to scale approximately like 1.5 times (number of flights), and ranges from 100 to 2000 for the problems probed.

The reduced problem may in most cases be further separated into a set of independent subproblems, which can be solved one by one. Some of the composite flights will formally arrive at the same effective airport they started from. This does not pose a problem. Indeed, if the airport in question is the hub, such a single flight constitutes a separate (trivial) subproblem, representing an entire forced rotation. Typically, one of the subproblems will be much larger than the rest and will be referred to as the kernel problem; the remaining subproblems will be essentially trivial.

In the formalism below, we allow for the possibility that the problem to be solved has been reduced as described above, which means that flights may be composite.

4 Potts Encoding

A naive way to encode the crew scheduling problem would be to introduce Potts spins in analogy with what was done by Gislén, Söderberg, and Peterson (1989, 1992), where each event (lecture) is mapped onto a resource unit (lecture room, time slot). This would require a Potts spin for each flight to handle the mapping onto crews.

Since the problem consists of linking together sequences of (composite) flights such that proper rotations are formed, starting and ending at the hub, it appears more natural to choose an encoding where each flight i is mapped, via a Potts spin, onto the flight j to follow it in the rotation:

$$s_{ij} = \begin{cases} 1 & \text{if flight } i \text{ precedes flight } j \text{ in a rotation} \\ 0 & \text{otherwise} \end{cases}$$

where it is understood that j is restricted to depart from the (effective) airport where i arrives. In order to ensure that proper rotations are formed, each flight has to be mapped onto precisely one other flight. This restriction is inherent in the Potts spin, defined to have precisely one component "on":

$$\sum_j s_{ij} = 1. \tag{4.1}$$

In order to start and terminate rotations, two auxiliary flights are intro-

Figure 2: Expansion of the propagator P_{ij} (○) in terms of s_{ij}. A line represents a flight, and (•) a landing and take-off.

duced at the hub: a dummy arrival a, forced to be mapped onto every proper hub departure, and a dummy departure b, to which every proper hub arrival must be mapped. No other mappings are allowed at the hub. Thus, a acts as a source and b as a sink of rotations; every rotation starts with a and terminates with b. Both have zero duration and leg count, as well as associated wait times.

Global topological properties, leg counts, and durations of rotations, cannot be described in a simple way by polynomial functions of the spins. Instead, they are conveniently handled by means of a propagator matrix, **P**, derived from the Potts spin matrix **s** (including a, b components) as

$$\begin{aligned} P_{ij} &= \left((\mathbf{1}-\mathbf{s})^{-1}\right)_{ij} \\ &= \delta_{ij} + s_{ij} + \sum_k s_{ik}s_{kj} + \sum_{kl} s_{ik}s_{kl}s_{lj} + \sum_{klm} s_{ik}s_{kl}s_{lm}s_{mj} + \cdots \end{aligned} \tag{4.2}$$

A pictorial expansion of the propagator is shown in Figure 2. The interpretation is obvious: P_{ij} counts the number of connecting paths from flight i to j. Similarly, an element of the matrix square of P,

$$\sum_k P_{ik}P_{kj} = \delta_{ij} + 2s_{ij} + 3\sum_k s_{ik}s_{kj} + \cdots, \tag{4.3}$$

counts the total number of (composite) legs in the connecting paths, while the number of proper legs is given by

$$\sum_k P_{ik}L_kP_{kj} = L_i\delta_{ij} + s_{ij}\left(L_i + L_j\right) + \sum_k s_{ik}s_{kj}\left(L_i + L_k + L_j\right) + \cdots, \tag{4.4}$$

where L_k is the intrinsic number of single legs in the composite flight k. Thus, $L_{ij} \equiv \sum_k P_{ik}L_kP_{kj}/P_{ij}$ gives the average leg count of the connecting paths. Similarly, the average duration (flight plus waiting time) of the paths from i to j amounts to

$$T_{ij} \equiv \frac{\sum_k P_{ik}t_k^{(\mathrm{f})}P_{kj} + \sum_{kl} P_{ik}t_{kl}^{(\mathrm{w})}s_{kl}P_{lj}}{P_{ij}}, \tag{4.5}$$

where $t_i^{(\mathrm{f})}$ denotes the duration of the composite flight i, including the embedded waiting time. Furthermore, any closed loop (such as obtained if two

flights are mapped onto each other) will make **P** singular; for a proper set of rotations, $\det \mathbf{P} = 1$.

The problem can now be formulatet as follows. Minimize T^{ab} subject to the contraints:[1]

$$L_{ib} \leq L_{\max} \tag{4.6}$$

$$T_{ib} \leq T_{\max} \tag{4.7}$$

for every departure i at the hub.

5 Mean Field Approach

We use a mean field (MF) annealing approach in the search for the global minimum. The discrete Potts variables, s_{ij}, are replaced by continuous MF Potts neurons, v_{ij}. They represent thermal averages $\langle s_{ij} \rangle_T$, with T an artificial temperature to be slowly decreased (annealed), and have an obvious interpretation of probabilities (for flight i to be followed by j). The corresponding probabilistic propagator **P** will be defined as the matrix inverse of $\mathbf{1} - \mathbf{v}$.

The neurons are updated by iterating the MF equations

$$v_{ij} = \frac{\exp(u_{ij}/T)}{\sum_k \exp(u_{ik}/T)} \tag{5.1}$$

for one flight i at a time, by first zeroing the ith row of $\mathbf{v}$,[2] and then computing the relevant local fields u_{ij} entering equation 5.1 as

$$u_{ij} = -c_1 t_{ij}^{(w)} - c_2 P_{ji} - c_3 \sum_k v_{kj} - c_4 \Psi \left(T_{\text{rot}}^{(ij)} - T_{\max} \right) - c_5 \Psi \left(L_{\text{rot}}^{(ij)} - L_{\max} \right), \tag{5.2}$$

where j is restricted to be a possible continuation flight to i. In the first term, $t_{ij}^{(w)}$ is the waiting time between flight i and j. The second term suppresses closed loops, and the third term is a soft exclusion term, penalizing solutions where two flights point to the same next flight. In the fourth and fifth terms, $L_{\text{rot}}^{(ij)}$ stands for the total leg count and $T_{\text{rot}}^{(ij)}$ for the duration of the rotation if i were to be mapped onto j and amount to

$$L_{\text{rot}}^{(ij)} = L_{ai} + L_{jb} \tag{5.3}$$

$$T_{\text{rot}}^{(ij)} = T_{ai} + t_{ij}^{(w)} + T_{jb}. \tag{5.4}$$

[1] Minimizing thus minimizes the total wait time, since the total flight time is fixed.

[2] To eliminate self-couplings.

The penalty function Ψ, used to enforce the inequality constraints (Ohlsson et al., 1993; Tank & Hopfield, 1986), is defined by $\Psi(x) = x\Theta(x)$ where Θ is the Heaviside step function. Normally, the local fields u_{ij} are derived from a suitable energy function; however, for reasons of simplicity, some of the terms in equation 5.2 are chosen in a more pragmatic way.

After an initial computation of the propagator **P** from scratch, it is subsequently updated according to the Sherman-Morrison algorithm for incremental matrix inversion (Press, Flannery, Teukolsky, & Vettering, 1986). An update of the ith row of **v**, $v_{ij} \to v_{ij} + \delta_j$, generates precisely the following change in the propagator **P**:

$$P_{kl} \to P_{kl} + \frac{P_{ki} z_l}{1 - z_i} \tag{5.5}$$

$$\text{where} \quad z_l = \sum_j \delta_j P_{jl}. \tag{5.6}$$

Inverting the matrix from scratch would take $O(N^3)$ operations, while the (exact) scheme devised above requires only $O(N^2)$ per row.

As the temperature goes to zero, a solution crystallizes in a winner-takes-all dynamics: for each flight i, the largest u_{ij} determines the continuation flight j to be chosen.

6 Test Problems

In choosing test problems, our aim has been to maintain a reasonable degree of realism, while avoiding unnecessary complication and at the same time not limiting ourselves to a few real-world problems, where one can always tune parameters and procedures to get good performance. In order to accomplish this, we have analyzed two typical real-world template problems obtained from a major airline: one consisting of long-distance (LD) and the other of short- and medium-distance (SMD) flights. As can be seen in Figure 3, LD flight time distributions are centered around long times, with a small hump for shorter times representing local continuations of long flights. The SMD flight times have a more compact distribution.

For each template we have made a distinct problem generator producing random problems resembling the template. A problem with a specified number of airports and flights is generated as follows. First, the distances (flight times) between airports are chosen randomly from a suitable distribution. Then a flight schedule is built up in the form of legal rotations starting and ending at the hub. For every new leg, the waiting time and the next airport are randomly chosen in a way designed to make the resulting problems statistically resemble the respective template problem.

Due to the excessive time consumption of the available exact methods (e.g., branch and bound), the performance of the Potts approach cannot be

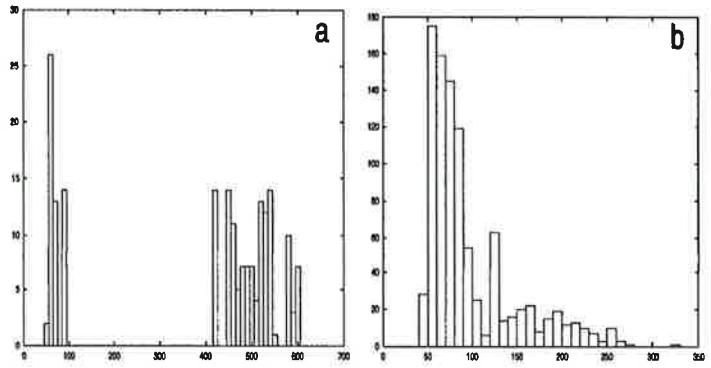

Figure 3: Flight time distributions in minutes for (a) LD and (b) SMD template problems.

tested against these—except for in this context ridiculously small problems, for which the Potts solution quality matches that of an exact algorithm.

For artificial problems of a more realistic size, we circumvent this obstacle. Since problems are generated by producing a legal set of rotations, we add in the generator a final check that the implied solution yields $T_{\min}^{\text{wait}}$; if not, a new problem is generated. Theoretically, this might introduce a bias in the problem ensemble; empirically, however, no problems have had to be redone. Also the two real problems turn out to be solvable at $T_{\min}^{\text{wait}}$.

Each problem is then reduced as described above (using a negligible amount of computer time), and the kernel problem is stored as a list of flights, with all traces of the generating rotations removed.

7 Results

We have tested the performance of the Potts MF approach for both LD and SMD kernel problems of varying sizes. As an annealing schedule for the serial updating of the MF equations, 5.1 and 5.5), we have used T_n=kT_{n-1} with $k = 0.9$. In principle, a proper value for the initial temperature T_0 can be estimated from linearizing the dynamics of the MF equations. We have chosen a more pragmatic approach: the initial temperature is assigned a tentative value of 1.0, which is dynamically adjusted based on the measured rate of change of the neurons until a proper T_0 is found. The values used for the coefficients c_i in equation 5.2 are chosen in an equally simple and pragmatic way: $c_1 = 1/\text{period}$, $c_2 = c_3 = 1$, while $c_4 = 1/\langle T^{\text{rot}} \rangle$ and $c_5 = 1/\langle L^{\text{rot}} \rangle$, where $\langle T^{\text{rot}} \rangle$ is the average duration (based on $T_{\min}^{\text{wait}}$) and $\langle L^{\text{rot}} \rangle$ the

Table 1: Average Performance of the Potts Algorithm for LD Problems.

N_f	N_a	$\langle N_f^{\text{eff}} \rangle$	$\langle N_a^{\text{eff}} \rangle$	⟨R⟩	⟨CPU time⟩
75	5	23	8	0.0	0.0 sec
100	5	50	17	0.0	0.2 sec
150	10	55	17	0.0	0.3 sec
200	10	99	29	0.0	1.3 sec
225	15	84	26	0.0	0.7 sec
300	15	154	46	0.0	3.4 sec

Notes: The superscript "eff" refers to the kernel problem; subscripts "f" and "a" refer to flight and airport, respectively. The averages are taken with ten different problems for each N_f. The performance is measured as the difference between the waiting time in the Potts and the local solutions divided by the period. The CPU time refers to DEC Alpha 2000.

average leg count per rotation, both of which can be computed beforehand. It is worth stressing that these parameter settings have been used for the entire range of problem sizes probed.

When evaluating a solution obtained with the Potts approach, a check is done as to whether it is legal (if not, a simple postprocessor restores legality; this is only occasionally needed); then the solution quality is probed by measuring the excess waiting time R,

$$R = \frac{T_{\text{Potts}}^{\text{wait}} - T_{\text{min}}^{\text{wait}}}{\text{period}}, \tag{7.1}$$

which is a nonnegative integer for a legal solution.

For a given problem size, as given by the desired number of airports N_a and flights N_f, a set of ten distinct problems is generated. Each problem is subsequently reduced, and the Potts algorithm is applied to the resulting kernel problem. The solutions are evaluated, and the average R for the set is computed. The results for a set of problem sizes ranging from $N_f \simeq 75$ to 1000 are shown in Tables 1 and 2; for the two real problems, see Table 3.

The results are quite impressive. The Potts algorithm has solved all problems, and with a very modest CPU time consumption, of which the major part goes into updating the P matrix. The sweep time scales like $(N_f^{\text{eff}})^3 \propto N_f^3$, with a small prefactor, are due to the fast method used (see equations 5.5 and 5.6). This should be multiplied by the number of sweeps needed—empirically between 30 and 40, independent of problem size.[3]

[3] The minor apparent deviation from the expected scaling in Tables 1, 2, and 3 is due

Table 2: Average Performance of the Potts Algorithm for SMD Problems.

N_f	N_a	$\langle N_f^{\text{eff}} \rangle$	$\langle N_a^{\text{eff}} \rangle$	$\langle R \rangle$	$\langle$CPU time$\rangle$
600	40	280	64	0.0	19 sec
675	45	327	72	0.0	35 sec
700	35	370	83	0.0	56 sec
750	50	414	87	0.0	90 sec
800	40	441	91	0.0	164 sec
900	45	535	101	0.0	390 sec
1000	50	614	109	0.0	656 sec

Note: The averages are taken with ten different problems for each N_f. Same notation as in Table 1.

Table 3: Average Performance of the Potts Algorithm for Ten Runs on the Two Real Problems.

N_f	N_a	$\langle N_f^{\text{eff}} \rangle$	$\langle N_a^{\text{eff}} \rangle$	$\langle R \rangle$	$\langle$CPU time$\rangle$	type
189	15	71	24	0.0	0.6 sec	LD
948	64	383	98	0.0	184 sec	SMD

Note: Same notation as in Table 1.

8 Summary

We have developed a mean field Potts approach for solving resource allocation problems with a nontrivial topology. The method is applied to airline crew scheduling problems resembling real-world situations.

A novel key feature is the handling of global entities, sensitive to the dynamically changing "fuzzy" topology, by means of a propagator formalism. Another important ingredient is the problem size reduction achieved by airport fragmentation and flight clustering, narrowing down the solution space by removing much of the suboptimal part.

High-quality solutions are consistently found throughout a range of problem sizes without having to fine-tune the parameters, with a time consumption scaling as the cube of the problem size. The basic approach should be easy to adapt to other applications (e.g., communication routing).

to an anomalous scaling of the Digital DXML library routines employed; the number of elementary operations does scale like N_f^3.

References

Durbin, R., & Willshaw, D. (1987). An analog approach to the traveling salesman problem using an elastic net method. *Nature, 326*, 689.

Gislén, L., Söderberg, B., & Peterson, C. (1989). Teachers and classes with neural networks. *International Journal of Neural Systems, 1*, 167.

Gislén, L., Söderberg, B., & Peterson, C. (1992). Complex scheduling with Potts neural networks. *Neural Computation, 4*, 805.

Hopfield, J. J., & Tank, D. W. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics, 52*, 141.

Ohlsson, M., Peterson, C., & Söderberg, B. (1993). Neural networks for optimization problems with inequality constraints—the knapsack problem. *Neural Computation, 5*, 331.

Peterson, C., & Söderberg, B. (1989). A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems, 1*, 3.

Press, W. P., Flannery, B. P., Teukolsky, S. A., & Vettering, W. T. (1986). *Numerical recipes: The art of scientific computing*. Cambridge: Cambridge University Press.

Tank, D. W., & Hopfield, J. J. (1986). Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems, CAS-33*, 533.

Received May 20, 1996; accepted November 27, 1996.