LU TP 96-6
May 10, 1996

Airline Crew Scheduling with Potts Neurons

Martin Lagerholm[1], Carsten Peterson[2] and Bo Söderberg[3]

Department of Theoretical Physics, University of Lund
Sölvegatan 14A, S-223 62 Lund, Sweden

Submitted to *Neural Computation*

Abstract:

A Potts feedback neural network approach for finding good solutions to resource allocation problems with a non-fixed 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/departure structure at the single airports.

[1] martin@thep.lu.se
[2] carsten@thep.lu.se
[3] bs@thep.lu.se

Introduction

Feedback neural network have in the last decade emerged as a useful method to obtain good approximate solutions to various resource allocation problems [1, 2, 3, 4]. Most applications have concerned fairly academic problems like the traveling salesman problem (TSP) and various graph partition problems [1, 2]. In refs. [3, 4] high school scheduling was approached. 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 [5], and a "purely" neural one. The template approach is advantageous e.g. for low-dimensional geometrical problems like the TSP, whereas for generic resource allocation problems, a purely neural Potts encoding is preferable.

A very 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 in a connected sequence of flights (*legs*), starting and ending at a 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 e.g. high school scheduling [3, 4] in the existence of non-trivial topological restrictions. A similar structure occurs in multi-task phone routing.

A common approach to this problem consists in converting it into a *set covering problem*, by (1) generating a large pool of legal rotation templates, and (2) 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 [6]. A disadvantage with this method is that the rotation generation for computational reasons has to be non-exhaustive for a large problem; thus, only a fraction of the solution space is available.

The approach to the crew scheduling problem developed in this letter 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 sub-optimal solutions. Then, a mean field annealing approch based on Potts neurons is applied, where a novel key ingredient is the use of a propagator formalism for handling topology, leg-counting, etc.

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.

Nature of 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 closed loop – *rotation* – starting and ending at the hub (see fig. 1).

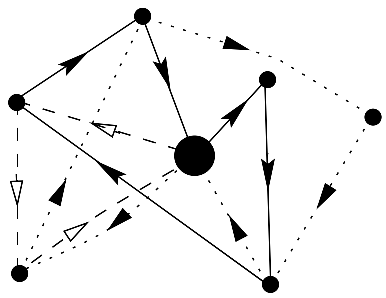

Figure 1: Schematic view of the 3 crew rotations starting and ending in a hub.

- The number of flight legs in a rotation must not exceed a given upper bound.
- The total duration (flight + waiting time) of a rotation is similarly bounded.

These are the crucial and difficult constraints; in a real-world problem there are often some 20 additional ones, which we for simplicity neglect; they constitute no additional challenge from an algorithmic point of view.

Without the above constraints, the problem would reduce to the *local problem* of minimizing waiting times independently at each airport; this can be done exactly in polynomial time. It is the global structural requirements that make the crew scheduling problem a challenge.

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

$$t_{ij}^{(\mathrm{w})} = \left(t_j^{(\mathrm{dep})} - t_i^{(\mathrm{arr})}\right) \bmod \mathrm{period}, \tag{1}$$

the total waiting time for a given problem can only change by an integer times the period. By demanding a minimal waiting time, the local problem (neglecting the global constraints) at each airport typically can be split up into independent subproblems, each containing a subset of the arrivals and an equally large subset of the departures. Some of these are trivial, forcing the crew of an arrival to continue to a particular departure. The minimal total wait-time for the local problem is straight-forward to compute, and will be denoted by $T_{\mathrm{min}}^{\mathrm{wait}}$.

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

- *Airport fragmentation*: Divide each airport into *effective airports* corresponding to the non-trivial 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 sub-optimal 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 size of the solution space, 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*, that 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*, while 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.

Potts Encoding

A naive way to encode the crew scheduling problem would be to introduce Potts spins in analogy with what was done in refs. [3, 4] 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 in linking together sequences of (composite) flights such that closed loops are formed, 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 be restricted to depart from the (effective) airport where i arrives. In order to ensure that proper closed loops are formed, each flight has to be mapped onto precisely one other flight (or terminate a rotation, in which case it is formally mapped on a dummy flight available only at the hub). This restriction is inherent in the Potts spin, defined to have precisely one component "on":

$$\sum_j s_{ij} = 1 \tag{2}$$

Global topological properties, leg-counts and durations of rotations, etc., cannot be described in a simple way by polynomial functions of the spins. Instead, they are conveniently handled by means

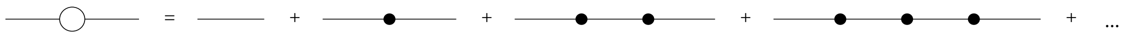

Figure 2: Expansion of the propagator P_{ij} ($\bigcirc$) in terms of s_{ij}. A line represents a flight, and ($\bullet$) a touch-down.

of a *propagator* matrix, $\mathbf{P}$, defined a in terms of the Potts spin matrix $\mathbf{s}$ by

$$P_{ij} = \left((\mathbf{1} - \mathbf{s})^{-1}\right)_{ij} = \delta_{ij} + s_{ij} + \sum_k s_{ik} s_{kj} + \sum_{kl} s_{ik} s_{kl} s_{lj} + \sum_{klm} s_{ik} s_{kl} s_{lm} s_{mj} + \ldots \tag{3}$$

A pictorial expansion of the propagator is shown in fig. 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,

$$\sum_k P_{ik} P_{kj} = \delta_{ij} + 2 s_{ij} + 3 \sum_k s_{ik} s_{kj} + \ldots \tag{4}$$

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

$$\sum_k P_{ik} L_k P_{kj} = L_i \delta_{ij} + s_{ij} \left(L_i + L_j\right) + \sum_k s_{ik} s_{kj} \left(L_i + L_k + L_j\right) + \ldots \tag{5}$$

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

$$T_{ij} \equiv \frac{\sum_k P_{ik} t_k^{(\mathrm{f})} P_{kj} + \sum_{kl} P_{ik} t_{kl}^{(\mathrm{w})} s_{kl} P_{lj}}{P_{ij}} \tag{6}$$

where $t_i^{(\mathrm{f})}$ denotes the duration of the composite flight i, including the embedded waiting time.

Furthermore, any improper loops (such as obtained e.g. if two flights are mapped onto each other) will make $\mathbf{P}$ singular – for a proper set of rotations, $\det \mathbf{P} = 1$.

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 $< s_{ij} >_T$, 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 $\mathbf{P}$ will be defined as the matrix inverse of $\mathbf{1} - \mathbf{v}$.

The neurons are updated by iterating the MF equations

$$v_{ij} = \frac{\exp(u_{ij}/T)}{\sum_k \exp(u_{ik}/T)} \tag{7}$$

for one flight i at a time, by first zeroing the i:th row of $\mathbf{v}$, and then computing the relevant *local fields* u_{ij} entering eq. (7) as

$$u_{ij} = -c_1 t_{ij}^{(\mathrm{w})} - c_2 P_{ji} - c_3 \sum_k v_{kj} - c_4 \Psi\left(T_{\mathrm{rot}}^{(\mathrm{ij})} - T_{\mathrm{rot}}^{\mathrm{max}}\right) - c_5 \Psi\left(L_{\mathrm{rot}}^{(\mathrm{ij})} - L_{\mathrm{rot}}^{\mathrm{max}}\right) \tag{8}$$

where j is restricted to be a possible continuation flight to i. In the first term, $t_{ij}^{(\mathrm{w})}$ is the local waiting time between flight i and j. The second term suppresses improper 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, $L_{\mathrm{rot}}^{(\mathrm{ij})}$ stands for the total leg count and $T_{\mathrm{rot}}^{(\mathrm{ij})}$ for the duration of the rotation if i where to be mapped onto j, and amount to

$$L_{\mathrm{rot}}^{(\mathrm{ij})} = L_{ai} + L_{jb} \tag{9}$$

$$T_{\mathrm{rot}}^{(\mathrm{ij})} = T_{ai} + t_{ij}^{(\mathrm{w})} + T_{jb} \tag{10}$$

Here, a and b are auxiliary dummy-flights (of zero duration and intrinsic leg count) representing the start/end of a rotation – at the hub, a is formally mapped onto every departure, and every arrival is mapped onto b. The penalty function Ψ, used to enforce the inequality constraints [6], 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 eq. (8) are chosen in a more pragmatic way.

After an initial computation of the propagator $\mathbf{P}$ from scratch, it is subsequently updated according to the Sherman-Morrison algorithm for incremental matrix inversion [7]. An update of the i:th row of $\mathbf{v}$, $v_{ij} \to v_{ij} + \delta_j$, generates precisely the following change in the propagator $\mathbf{P}$:

$$P_{kl} \to P_{kl} + \frac{P_{ki}\, z_l}{1 - z_i} \tag{11}$$

$$\text{where} \quad z_l = \sum_j \delta_j\, P_{jl} \tag{12}$$

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

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), the other of short/medium distance (SMD) flights. As can be seen from fig. 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.

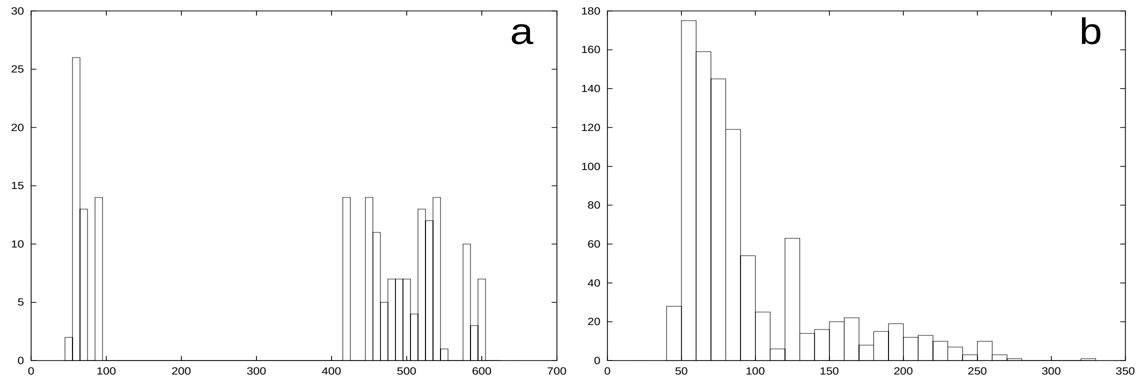

Figure 3: Fligh time distributions in minutes for **(a)** LD and **(b)** SMD template problems.

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, the performance of the Potts approach cannot be 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 more realistic size we circumvent this obstacle in the following way: since problems are generated by producing a legal set of rotations, we add in the generator a final check that the implied solution yields $T_{\rm min}^{\rm wait}$; 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 $T_{\rm min}^{\rm wait}$.

Each problem then is 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.

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 eqs. (7,11), 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 eq. (8) are chosen in an equally simple and pragmatic way: $c_1 = 1/\text{period}$, $c_2 = c_3 = 1$, while $c_4 = 1/ < T^{\rm rot} >$ and $c_5 = 1/ < L^{\rm rot} >$, where $< T^{\rm rot} >$ is the average duration (based on $T_{\rm min}^{\rm wait}$) and $< L^{\rm rot} >$ the 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

N_f	N_a	$< N_f^{\rm eff} >$	$< N_a^{\rm eff} >$	< 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

Table 1: Average performance of the Potts algorithm for **LD** problems. The superscript "eff" refers to the kernel problem, subscript "f" and "a" refers to flight respectively airport. The averages are taken with 10 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.

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 post-processor restores legality – this is only occasionally needed), then the solution quality is probed by measuring the excess waiting time R,

$$R = \frac{T^{\rm wait}_{\rm Potts} - T^{\rm wait}_{\rm min}}{\rm period}, \tag{13}$$

which is a non-negative 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 10 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.

N_f	N_a	$< N_f^{\rm eff} >$	$< N_a^{\rm eff} >$	< R >	< CPU time >
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

Table 2: Average performance of the Potts algorithm for **SMD** problems. The averages are taken with 10 different problems for each N_f. Same notation as in table 1.

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

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

Table 3: Average performance of the Potts algorithm for 10 runs on the two **real** problems. Same notation as in table 1.

sweep time scales like $(N_f^{\rm eff})^3 \propto N_f^3$, with a small prefactor, due to the fast method used, eqs. (11, 12). This should be multiplied by the number of sweeps needed – empirically between 30 and 40, independently of problem size[4].

Summary

We have developed a mean field Potts approach for solving resource allocation problems with a non-trivial 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 sub-optimal 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, like e.g. communication routing.

References

[1] J.J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics* **52**, 141 (1985).

[2] C. Peterson and B. Söderberg, "A New Method for Mapping Optimization Problems onto Neural Networks", *International Journal of Neural Systems* **1**, 3 (1989).

[3] L. Gislén, B. Söderberg and C. Peterson, "Teachers and Classes with Neural Networks", *International Journal of Neural Systems* **1**, 167 (1989).

[4] L. Gislen, B. Söderberg and C. Peterson, "Complex Scheduling with Potts Neural Networks", *Neural Computation* **4**, 805 (1992).

[5] R. Durbin and D. Willshaw, "An Analog Approach to the Traveling Salesman Problem using an Elastic Net Method". *Nature* **326**, 689 (1987).

[4]The minor apparent deviation from the expected scaling in tables 1, 2 and 3 are due to an anomalous scaling of the Digital DXML library routines employed; the number of elementary operations does scale like N_f^3.

[6] M. Ohlsson, C. Peterson and B. Söderberg, "Neural Networks for Optimization Problems with Inequality Constraints - the Knapsack Problem", *Neural Computation* **5**, 331 (1993).

[7] See e.g. W.P. Press, B.P Flannery, S.A. Teukolsky and W.T. Vettering, *Numerical Recipes, The Art of Scientific Computing*, Cambridge University Press, Cambridge (1986).