Typewriter Keyboards via Simulated Annealing

Lissa W. Light Peter G. Anderson*

In *AI Expert*, September, 1993

Abstract

We apply the simulated annealing algorithm to the combinatorial optimization problem of typewriter keyboard design, yielding nearly optimal key-placements using a figure of merit based on English letter pair frequencies and finger travel-times. Our keyboards are demonstrably superior to both the ubiquitous QWERTY keyboard and the less common Dvorak keyboard.

The paper is constructed as follows: first we discuss the historical background of keyboard design; this includes August Dvorak's work, and a "figure-of-merit" (scalar) metric for keyboards. We discuss a theory of keyboard designs: why keyboard design is a combinatorial problem, how combinatorial problems are typically solved, what is simulated annealing, and why it is especially suitable for the problem at hand. Next we discussed the results, and compare the keyboards produced by simulated annealing to QWERTY and Dvorak's keyboard. Finally, we suggest some future lines of inquiry.

Keyboard Design

Since the invention of the typewriter in the late nineteenth century, there has been an interest in maximally efficient keyboards. The original, and most widely used, typewriter keyboard is the "QWERTY" keyboard—named after the first six characters in the top row of letters—or the Sholes keyboard—after its inventor. In its inception, the keyboard was designed to ease typing for the hunt-and-peck two-fingered typist; touch typing was not introduced until the early 1900's. Early typewriters were constructed of slow-moving mechanical parts, and rapid typing would jam the keys. QWERTY was designed to *slow down* typists ([21], p. 266)! QWERTY is still the most widely used keyboard, so these archaic design decisions have brought about several unhappy consequences:

- QWERTY distributes typing unequally among the hands, favoring the left hand over the right. Logic would seem to dictate the reverse, since the majority of typists are right handed, and the right hand is typically stronger.

- QWERTY overloads the weaker fingers, overlooking the fact that the index and middle fingers are stronger than the fourth and fifth fingers.

- QWERTY uses the home (middle) row of keys less than one-third of the time—it should instead *maximize* the use of this row of keys to *minimize* finger travel time.

*Computer Science Department, Rochester Institute of Technology, Rochester, NY 14623-0887; internet: pga@cs.rit.edu

- QWERTY seems to ignore the fact that alternate hand typing of common letter pairs is by far faster than same hand typing them ([7] and [21], p. 267).

Many alternatives to QWERTY have been proposed over the years: these include the "Combinational Keyboard" by Nelson in 1920, the "Orthographic Keyboard" by Gilbert in 1930, and, most importantly, the Dvorak Simplified Keyboard (or the D.S.K.) by August Dvorak in 1932. Each of these attempted to overcome some of the difficulties inherent in QWERTY, but only Dvorak's has had any impact upon the manufacture and production of typewriting equipment. Today, there are a number of typewriters that offer Dvorak keyboards, and there are several software programs that remap the QWERTY character set to Dvorak's permutation; nevertheless, the Dvorak keyboard remains a weak competitor to QWERTY.

Dvorak designed his keyboard in the 1930's. He realized that the standard keyboard was slow and awkward to use. He was interested in an "ergonomically designed" keyboard that eased typing by minimizing distant keystrokes, preventing clumsy finger motions, and maximizing use of the strong index and middle fingers. Dvorak carefully studied the process of typing and the letter frequencies and sequences in English, and he formulated several principles of keyboard design, most of which are still accepted by keyboard designers and theorists today.

Dvorak's keyboard geometry differs lightly from the QWERTY: he put four more keys on the right.

Among Dvorak's objectives were [5]:

- A keyboard designed around a statistical analysis of common English letter pairs.
- A keyboard that would decrease the likelihood of error by making typing more "natural" and less awkward, by reducing finger motions from row to row.
- A keyboard that would maximize the use of the "home" (middle) row and the fastest sorts of keystrokes (those that use alternate-hand typing, rather than same-hand typing).
- A keyboard that would give the strong right hand more work to do.
- A keyboard that is easier to learn, less subject to error, and less tiring.

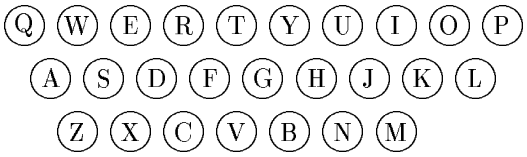


Figure 1: The famous QWERTY keyboard. Its cost is 1542.

Underlying these objectives are several assumptions made by Dvorak and other keyboard designers (including the present authors). Letter and letter-pair data are for English; other languages would presumably require other keyboards. And the character set supported is limited to the 26 letters of the English language and does not include characters in other languages (e.g., œ, ø, ñ, ß, etc.). Keyboard geometry, with three rows of letters, is accepted as given, as are the assumptions that every particular character occurs but once on the keyboard and is struck by one particular finger, and that the space bar is struck by the thumb.

Taking these factors into account, QWERTY and Dvorak's keyboards differ substantially (Figures 1 and 2).

Many experimental efforts have attempted to compare the two keyboard designs ([21],

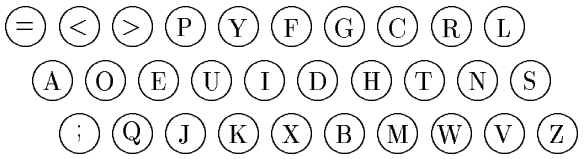


Figure 2: Dvorak's keyboard. Its cost is 1502. (We extrapolated the travel time data for the extra keys on the right.)

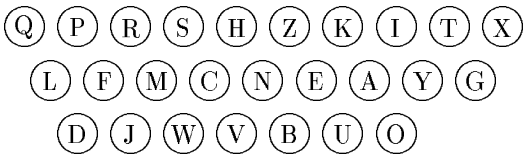


Figure 3: Our best keyboard. Its cost is 1428.

p. 271 and [29]), and a recent attempt by Kinkead has suggested a simple figure of merit, "cost," for any particular configuration. We use Kinkead's method in this paper.

Given a keyboard design, it is simple to measure its "cost." Just three pieces of information are required: the relative frequency, $F_{\alpha\beta}$, of every letter pair, (α, β), for English (where $\alpha\beta = AA, AB, \ldots, ZY, ZZ$), keying times (or "travel times") between positions on the standard three-row keyboard, T_{pos_1, pos_2}, and the positions, *pos(α)*, of every letter, α, on the keyboard. The frequency of every possible two-letter combination is multiplied by its "travel time" (the time to get from the first letter to the second) between the positions occupied by the two letters, α and β; and the results are summed. This sum is the "cost," a single value that represents the figure of merit of a particular keyboard design—keyboard designers should seek to minimize this cost function. Cost should, roughly, be proportional to *words per minute*.

$$Cost = \sum_{\alpha=A}^{Z} \sum_{\beta=A}^{Z} F_{\alpha\beta} T_{pos(\alpha), pos(\beta)} \qquad (1)$$

Figures 1, 2, and 3 show the costs associated with QWERTY, Dvorak's keyboard, and the best (least cost) keyboard we have been able to discover—its cost is 1428. A lower bound for the cost in equation (1), obtained by associating the most frequent letter pairs with the fastest travel times, is 1271. But that low bound is based on ignoring the meaning of the terms in the formula, and is clearly impossible.

At this point, the methodology used here must be justified. The conclusions of this paper rest, of course, upon the validity of the assumptions and the data that have been used. An important piece of this data is travel time information. We assumed that there are static travel times among various keyboard positions, regardless of the specific letters that occupy such positions. This is a controversial assumption. Some commentators argue that the travel time computed for a QWERTY machine may in fact be higher than for a Dvorak machine, for this latter keyboard was specifically designed to minimize fatigue and awkward motions [25]. Cassingham ([4], p. 1) argues the reverse: typing speed is wholly determined by such factors as finger strength, finger movements, and keyboard configuration. Under this interpretation, travel times are static values and do not change as a function of the keyboard being used [15]. We believe that there is insufficient experimental evidence for the former position, and there is at least prima facie evidence for the latter. There has not yet been a study that shows the influence of keyboard type on travel time. In the absence of such evidence, we use the simpler hypothesis.

Kinkead's work in establishing travel times between keys was limited to the conventional QWERTY configuration, and the only times that he was able to measure were between letter locations. He did not include the semicolon key (on the far right position of the home row)

nor the slash key (on the far right position of the bottom row). Because Dvorak used these keys for letters, Kinkead's data is not adequate for a completely accurate analysis of keyboard types. To mitigate this deficiency, we extrapolated the missing travel times from the given data. Though these extrapolated times are only approximations, they do make a comparison between the QWERTY keyboard and the Dvorak keyboard possible.

Keyboard Design as Combinatorial Optimization

A computer scientist's viewpoint is that the problem of optimal keyboard design is a problem of combinatorial optimization. The size of the set of all possible keyboards is the factorial of the number of keys to be placed. For just the 26 letters, we have:

$$26! = 403,291,461,126,605,635,584,000,000.$$

If we consider digits and punctuators, we have $40! \approx 8 \times 10^{47}$ keyboards to consider. These astronomically large numbers preclude any exhaustive search of possible keyboards to locate the best—even by the fastest imaginable computer.

Problems like this, with a search space that grows exponentially with the size of the problem, are often *NP-complete problems*, for which "...No method for exact solution with a computing effort bounded by a power of N has been found..." ([16], p. 671). The *Traveling Salesman Problem* [18] is a paradigm case of such a problem: a salesman wants the minimum-length circuit to visit N cities. As N increases, the number of possible routes is $\frac{1}{2}N!$ which increases exponentially,[1] so it becomes impossible to enumerate all possible solutions in a reasonable time. Problems for which are the time to solve them is an exponential (or worse) function of the length of the problem statement are called "intractable." It is straightforward to recast our keyboard optimization problem as a traveling salesman problem and vice versa, which shows that *our* problem is also NP-complete.

Although the NP-complete problems are widely believed to be intractable, many algorithms do exist for finding nearly optimal solutions to them. The simplest approach—although this could hardly be called an algorithm—is to repeatedly generate solutions (using a pseudo random number generator), evaluate their merit, and select the best one found when one's computing budget is exhausted. A more intelligent approach is *greedy improvement,* which gets its name from the observation that shortsighted greed may not be the best longterm policy. The greedy algorithm considers potential modifications to the present situation (keyboard layout, salesman's tour, etc.), which may be selected randomly or systematically (randomly may be preferred in case the search space is immense), and, if the proposed situation is an improvement over the present situation, the improvement is accepted. For typical NP-complete problems, greedy algorithms quickly lead to dead ends, which are called *local optima.* This is also known as "deterministic hill climbing" or "simple gradient descent" (the metaphor should be chosen to match a "merit" which we want increase, or a "cost" which we want to decrease).

For keyboard design, the greedy algorithm works as follows: start with a randomly generated keyboard, with each of the 26 positions of the keyboard occupied by a different letter. Compute the cost of this keyboard as in equation (1). Switch a pair of keys, determine the cost of the modified keyboard, and keep the keyboard with the lower cost. Repeat this

[1] Stirling's approximation for factorials gives: $N! \approx (\frac{N}{e})^N \sqrt{2\pi N}$

cycle of modify-and-evaluate until the cost no longer decreases. One can use a stopping rule based on a certain amount of patience: quit trying if no improvement is made in 1,000 attempts. There are only $\binom{26}{2} = 325$ possible key pairs to choose to interchange, so a systematic listing of all of these is reasonable.

Though this algorithm will converge to a keyboard layout that cannot be improved by a transposition of two keys, the final value of the function is merely locally optimal, rather than globally so. The computed cost will be minimal only in relation to the portion of the solution space that is being explored. Since the algorithm begins from a given starting point, and the generated permutations must travel downhill to be accepted, only a small set of possible keyboards are even considered, and these bear a family resemblance to one another. In technical terms, they belong to the same "neighborhood structure" ([1], p. 6); they reside within the same part of the topological structure that constitutes the solution space. Given the local nature of this greedy search process, only a small fraction of the search space is ever sampled.

One approach to mitigate this inevitable consequence of greedy algorithms is to construct several different starting solutions, and to travel downhill from each of these. The best of these local optimal is then accepted. The problem of local optima is not entirely avoided, however, because the final result is simply the most favorable of the local optima that have been computed.

Simulated annealing (SA) represents an important improvement over simple greedy algorithms. In a watershed paper, "Optimization by Simulated Annealing," Kirkpatrick, Gelatt, and Vecchi [16], showed that the problem of local optima can be at least partially mitigated by forcing the algorithm to examine larger portions of the possible solution space rather than engaging in mere local exploration. Simulated annealing samples many widely distant portions of the solution space, portions that bear little family resemblance to each other. SA is not, like gradient descent, invariably greedy; it does not always choose to lower the cost function. SA's behavior is governed stochastically, or probabilistically, so that in its initial exploration of possible keyboards it will willingly accept those with higher costs associated with them. As the procedure continues, however, the probability of accepting higher costs decreases with time, and eventually dwindles nearly to zero.

Kirkpatrick, *et al.* explain their procedure as an elaboration of the Metropolis algorithm, according to which stochastic principles of statistical mechanics are employed to avoid the local minima problem. A variable, T ("temperature"), regulates the amount of free searching that is done: initially, at a high temperature, there is a great deal of random searching around the solution space; as the temperature is lowered, the degree of searching is gradually constricted, and eventually curtailed almost entirely. This method allows a great portion of the solution space to be traversed, with the result that local minima are avoided, and close-to-optimal solutions are achieved. To use a different spatial metaphor, there is little chance that different neighborhoods will establish rival coalitions; instead, the entire structure will cooperatively support the same interpretation ([1], p. 131).

SA uses T to adjust the probability that a configuration will be accepted; this is inspired by the Boltzmann distribution of statistical mechanics: if two configurations have costs C_1 and C_2, then the probability of accepting the configuration with cost C_2 is given by

$$Pr[accept\ C_2] = (1 + e^{\frac{C_1 - C_2}{T}})^{-1}$$

For very high temperatures, T, the probabil-

ity is close to 0.5, so any two configurations are almost equally acceptable. For lower temperatures, the configuration with the lower cost is given a slight edge, and as T approaches zero, the probability of accepting the lower cost configuration approaches unity.

This annealing is analogous to the procedure chemists use to grow large crystals. To form a crystal with no defects, one with perfect lattices and symmetrical throughout, requires first that the temperature of the substance be raised to the melting point, and be lowered very slowly. If cooling is too rapid, the substances will exhibit mere metastable order: they will be only fragmentally (locally) ordered, and will have no global stability. Slow cooling, on the other hand, allows the atoms to redistribute symmetrically within the structure ([1], p. 13-14).

The analogy of combinatorial optimization problems to the process of crystalline growth is this: as the hills and valleys of the solution space are being investigated, there is, initially, a freedom to explore the entire configuration of possibilities. Locality of position is avoided at the outset. As the temperature decreases very gradually, so does the domain of exploration: it slowly becomes more centralized, but only when many other solutions have been sampled and rejected. Eventually, the solution converges to an optimum that is not merely local, though is not necessarily global. The solutions are frequently tagged "near-optimal."

Annealing Keyboards

It is straightforward to program a simulated annealing algorithm for keyboard design. The data structures and computational entities used are:

- An array with 26 locations is used to represent the keyboard: each index represents a position on the keyboard (from 0 to 25), and each value represents a letter of the alphabet (from A to Z but not necessarily in that order).
- The cost function parameters, letter-pair frequencies and key-position pair travel times.
- The probability function, which includes the variable T.
- A "cooling schedule," that determines how rapidly T is decreased.
- A transition mechanism that regulates how keyboard permutations are generated.

The program's structure is:

```
Initialize the keyboard
  (randomly or with user input).
Determine the present keyboard's cost.
Initialize T (very hot).

Repeat
  Repeat
    Choose two keys to transpose.
    Determine the modified keyboard's cost.
    If new_cost < old_cost, Then
      Accept the modified keyboard
    Else
      Use the Boltzmann distribution
        to determine acceptance.
  Until the cost stablizes
  Reduce T
Until T < minimum_temperature
```

This procedure continues in an iterative fashion: a new permutation is generated, its cost is computed, and its acceptance is evaluated. The entire process is regulated by the temperature variable, T, and by a Markov chain of permuted keyboards that are created.

T is initially set so that all candidate keyboards are almost equally acceptable to the system. The temperature is gradually decreased by an exponential decay rule such as:

$$T \leftarrow \rho T$$

where $0 < \rho < 1$ is the constant cooling rate. Typical values of ρ lie between 0.8 (rapid cooling) and 0.99 (slow cooling) ([1], p. 59). The slower the cooling schedule, the longer the process takes to determine a keyboard; the more rapid the cooling, the more likely we are to terminate the search prematurely with an inferior keyboard.

The cooling schedule is governed by the behavior of the Markov chains. As keyboard permutations are produced, they create chains of transitions, or Markov chains. This is a series of permuted keyboards at a given temperature. If the Boltzman probability equation is properly formulated, then at each temperature a point of equilibrium will be reached, so that many permutations are created without affecting the cost. At this point, the temperature is reduced, and the cycle continues until the temperature is close to zero.

In our experimentation we found that an initial temperature of 50.0 and $\rho = 0.99$ ensured the successful production of a keyboard. The length of the Markov chains, of course, varied at each temperature, increasing as the temperature decreased.

Results

The results of our study produce a good keyboard (Figure 3) that is unlike both QWERTY and Dvorak's, though it is more akin to the latter. After running the simulated annealing program several times, several good keyboards were produced, all of which had several features in common. (A sample run is shown in Figure 4.) In all cases, the cost of the our keyboards was consistently below both QWERTY and Dvorak's. QWERTY keyboard is by far the slowest; Dvorak's was significantly faster; and the one produced by SA was the fastest. Tables 1 and 2 summarize our results.

The Dvorak keyboard performs better than the QWERTY by a factor of 3.5% measured in terms of letter pairs, and the best keyboard our algorithm was able to discover is 8% faster than QWERTY.

Aside from raw speed, the optimal keyboard has other features that make it a good performer. Dvorak argued the most common digrams should be typed by alternate hands. According to our matrix of English digrams, the 5 most frequent digram pairs are shown in Table 1.

Pair	Freq	QWERTY	Dvorak	SA
TH	361	Y	N	Y
HE	330	Y	Y	Y
IN	240	N	Y	Y
ER	204	N	Y	Y
AN	196	Y	Y	Y
RE	180	N	Y	Y
ON	165	N	Y	Y
AT	143	N	Y	N
EN	140	Y	Y	Y
ND	131	Y	N	N

Table 1: Alternate hand typing of the top 10 English letter pairs. The frequency listed is out of 10,000 pairs, so these 10 constitute 18% of all pairs in English usage.

Like Dvorak's, our keyboard provides alternate hand typing for most of the most common English letter pairs.

Our keyboard also conforms to other, intuitive, rules of keyboard design:

- The most frequent letters in English—*E, T, A, O, I, N*—are mostly typed by the strong index and middle fingers.
- Letters that are rarely used—*W, X, Q, V*—are relegated to the bottom row or are typed by the weaker fingers.

To summarize: the optimal keyboard resulted in a significant time savings in typing the complete set of English language digrams. It also allows commonly used keys to be easily reached by the stronger fingers. Finally, it maximizes alternate hand typing of the most common letter pairs.

Future Lines of Inquiry

Throughout our investigation, many interesting issues concerning keyboard design and combinatorial optimization have arisen. We intend to pursue some of these in future work. Among these issues are: collecting valid travel time data (the lack of such data is of real concern to us), and using a more sophisticated cost function that would be based on larger pieces of English text, such as letter triples.

Another approach to this type of combinatorial optimization is *genetic algorithms* [11], a technique which has been successfully applied to traveling salesman and circuit layout problems.

References

[1] E. H. L. Aarts and J. H. M. Korst, *Simulated Annealing and Boltzmann Machines: a Stochastic Approach to Combinatorial Optimization and Neural Computing,* John Wiley & Sons, New York, 1989.

[2] D. G. Alden, R. W. Daniels, and A. F. Kanarick, "Keyboard design and operation: a review of the major issues," *Human Factors*, 14, pp. 275-293, 1972.

[3] Robert Baily, *Human Performance Engineering: Using Human Factors/Ergonomics to Achieve Computer System Usability,* Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[4] R. C. Cassingham, "Controvery over travel time," *Dvorak Development* , Vol. 13, No 3, Spring, 1986.

[5] R. C. Cassingham, *The Dvorak Keyboard: The Ergonomically Designed Typewriter, Now an American Standard,* Freelance Communications, Arcata, California, 1986.

[6] James H. Cervantes and Richard R. Hildebrant [1987], "Comparison of three neuron based computation schemes," *IEEE International Conference on Neural Networks (1st),* IEEE Service Center, Piscataway, N.J., 657-671, 1987.

[7] A. Dvorak, N. F. Merrick, W. L. Dealey, and G. C. Ford, *Typewriting behavior: Psychology Applied to Teaching and Learning Typewriting,* American Book Co., New York, 1936.

[8] Yoon-Pin Simon Foo and Yoshiyasu Takefuji, "Stochastic neural networks for solving job shop scheduling, part 1: problem representation," *IEEE International Conference on Neural Networks (2nd),* 2 (July), 275-280, 1988.

[9] Caxton C. Foster *Cryptanalysis for Microcomputers,* Hayden Book Company, Rochelle Park, New Jersey, 1982.

[10] J. G. Fox and R. G. Stansfield, "Digram keying times for typists," *Ergonomics,* 7, 317-320, 1964.

[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley, New York, 1989.

[12] Kurt M. Gutzmann, "Combinatorial optimization using a continuous state boltzmann machine," *IEEE International Conference on Neural Networks (1st),* IEEE Service Center, Piscataway, N.J., 721-734, 1987.

[13] Richard S. Hirsch, "Effects of standard versus alphabetical keyboard formats on typing performance," *Journal of Applied Psychology,* 54, 6, 484-490, 1970.

[14] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics,* 52, 141-152, 1985.

[15] Robin Kinkead, "Typing speed, keying rates, and optimal keyboard layouts," *Proceedings of the Human Factors Society* 19th Annual Meeting, 159-161, 1975.

[16] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecci, "Optimization by simulated annealing," *Science,* 220, 671-680, 1983.

[17] Edmund T. Klemmer, "Keyboard entry," *Applied Ergonomics,* 2, 1, 26, 1971.

[18] E. L. Lawler, A. H. G. Rinooy Khan, and D. B. Shmoys (eds.) *The Traveling Salesman Problem,* John Wiley & Sons, New York, 1985.

[19] J. L. McClelland and D. E. Rumelhart (eds.), *Explorations in Parallel Distributed Processing,* MIT Press, Cambridge, MA, 1988.

[20] Robert W. Monty, Harry L. Snyder and Gerald G. Birdwell, "Keyboard design: an investigation of user preference and performance," *Proceedings of the Human Factors Society 27th Annual Meeting,* 201-206, 1983.

[21] Jan Noyes, "The QWERTY keyboard: a review," *International Journal of Man Machine Studies,* 18, 265-281, 1983.

[22] J. Ramanujam and P. Sadayappan, "Optimization by neural networks," *IEEE International Conference on Neural Networks (2nd),* 2 (July), 325-332, 1988.

[23] D. E. Rumelhart and D. A. Norman, "Simulating a skilled typist," *Cognitive Science,* 6, 136, 1982.

[24] Timothy Salthouse, "The skill of typing," *Scientific American,* 250 (2), 128-135, 1984.

[25] R. Seibel, "Data entry devices and procedures," in *Human engineering guide to equipment design*, H. D. Van Cott and R. G. Kinkead (eds), Washington, DC, U. S. Government Printing Office, 1972.

[26] Sol Sherr (ed.), *Input Devices,* Academic Press, Old Chatham, New York, 1988.

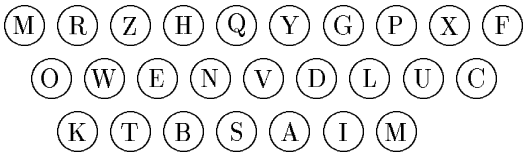
[27] Gene A. Tagliarini and Edward W. Page, "Solving constraint satisfaction problems with neural networks," *IEEE International Conference on Neural Networks (1st),* IEEE Service Center, Piscataway, N.J., 741-747, 1987.

[28] Bev and Bill Thompson, "Neurons, analog circuits, and the traveling salesperson," *AI Expert,* July, 1987.

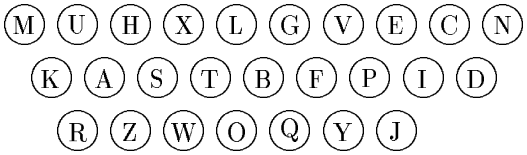
[29] H. Yamada, "A historical study of typewriters and typing methods: from the position of planning Japanese parallels," *J. of Info. Proc.*, 2, 175-202, 1980.

Keyboard	R : L %	Top	Home	Bottom
QWERTY	43 : 57	52	32	16
Dvorak	56 : 44	8	70	22
SimAnn	52 : 48	38	43	19

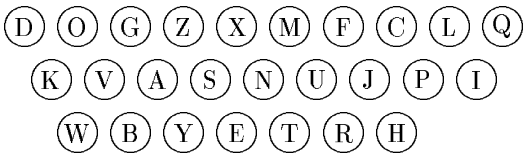
Table 2: Comparisons of three keyboards. "R : L" gives the percentage of letters typed by the right and left hands respectively. "TOP, HOME, BOTTOM" give the percentage of letters typed on each of these keyboard rows.



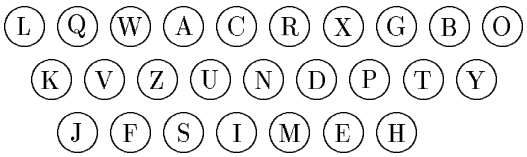
Cost is 1753.68.; temperature = 10.00.



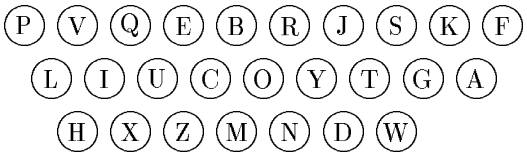
Cost is 1642.794; temperature = 9.01.



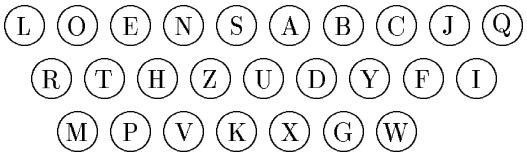
Cost is 1696.696; temperature = 8.01.



Cost is 1694.408; temperature = 7.01.

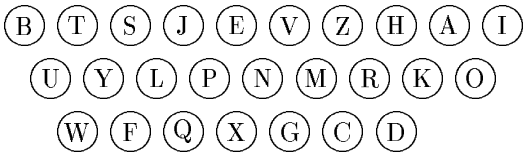


Cost is 1643.071; temperature = 6.01.

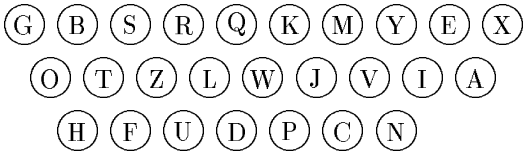


Cost is 1667.289; temperature = 5.01.

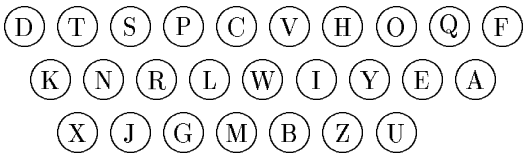
Figure 4: A sample run.



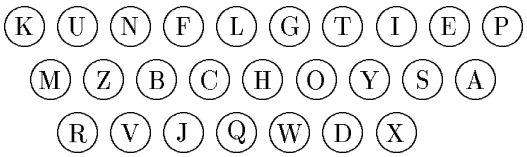
Cost is 1577.014; temperature = 4.01.



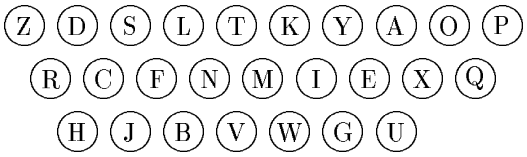
Cost is 1581.364; temperature = 3.01.



Cost is 1544.128; temperature = 2.01.



Cost is 1503.293; temperature = 1.01.



Final keyboard cost is 1429.168; temperature = 0.01.

Figure 5: A sample run, continued.