HEURISTIC ACCELERATION OF FORCE-DIRECTED PLACEMENT

Rob Forbes

Hewlett-Packard Company

Abstract

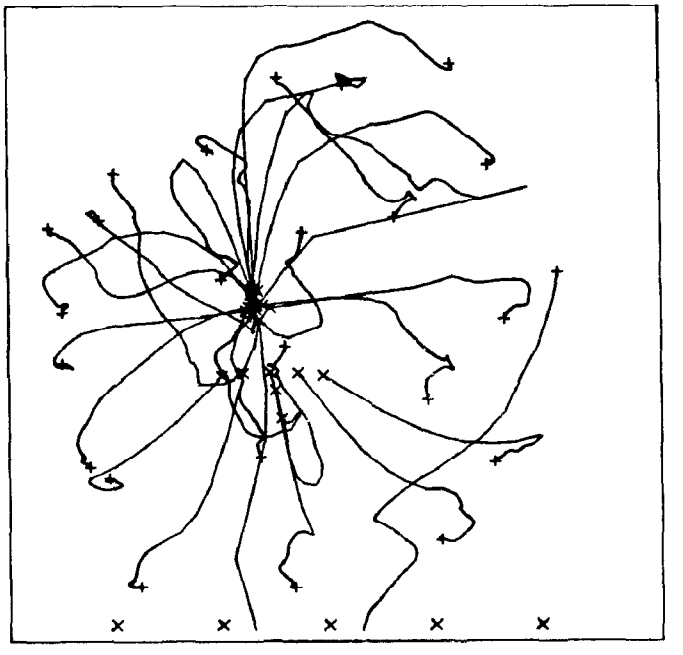
Two heuristic methods are presented for accelerating convergence of a force-directed placement problem. The first stabilizes the derivative of the repulsion force. The second uses information on device movement and instability characteristics to make a predictive extrapolation. Convergence is accelerated by replacing iterations with the faster heuristic iterations. A standard implementation from the literature is made three to four times faster by using these techniques.

Introduction

This paper is concerned with accelerating the calculations in a standard method of device placement. This is done in two independent ways. The first involves identifying and reducing a source of instability in the equations governing the problem, and will be discussed below under Problem Formulation. The second involves use of an extrapolation. Both are heuristics.

A heuristic, as used here, is an approximation which usually makes it possible to find a satisfactory solution in much less time than would be required by a more exact method. For instance, the second of the methods to be presented here uses an approximation derived from the successive positions of a device during the solution procedure. The path a device has followed during previous iterations is used to predict its position after one or more future iterations. Figure 1 shows the paths followed by devices during an iterative solution. Note how smooth the curves are--this suggests that extrapolations will be accurate most of the time.

The primary goal of device placement is an arrangement of devices that can be routed (connected) in the limited amount of board area available. Force-directed placement (FDP) increases routability by placing strongly connected devices close together. This minimizes total connection length, making it more likely that there will be enough area to make all connections.



x - initial position + - final position

Figure 1. Device paths during iteration

FDP is done by applying a physical model, where each pair of electrically connected devices is considered to be joined by a spring whose stiffness is proportional to the number of connections between the devices. Iterative solution is done by letting the devices move in response to the spring forces. Movement is toward an equilibrium location where the forces on each device are balanced, at which point movement ceases.

Unless there happen to be devices on the margin of the board, the movable devices tend to collapse into a cluster due to their mutual attraction. This does not leave room for routing. To prevent this clustering it is necessary to add a counter-force. A repulsion force between neighboring devices provides the necessary spreading effect.

Because the repulsion force tends to inhibit changes in the relative position of the devices, in practice the introduction of the repulsion force is delayed for a short time to establish

relative positions. The devices in Figure 1 are spreading from a cluster as a result of the introduction of repulsion.

One consequence of this method is that each iteration requires a significant amount of calculation to determine the force on each device. Depending on the degree of interconnectivity and the details of the implementation, this can lead to a worst case computation time proportional to the square of the number of devices. This results in a rapid increase in required computation time as a function of problem size.

The method in [1] is separated into Phases I and II. Phase I places point devices on a continuous plane. Phase II considers actual device size and placement on a grid. Since devices are modeled as points in Phase I, the product of Phase I is a set of ideal locations, without regard for device overlap. These locations are then used by Phase II as the basis for a physically realizable placement.

This paper is concerned with a more efficient solution to Phase I. Phase II is not addressed here. It has been suggested ([1]) that Phase I can be modified to address some Phase II issues (e.g., incorporating device size information in repulsion).

Prior Work

The physical model of devices moving in a continuous space rather than a grid appeared in [1], and was significantly improved in [2]. The choice between numeric solution methods has been made largely for computational efficiency. Variations of the well-known Newton-Raphson (NR) method [3] are used in both [1] and [2].

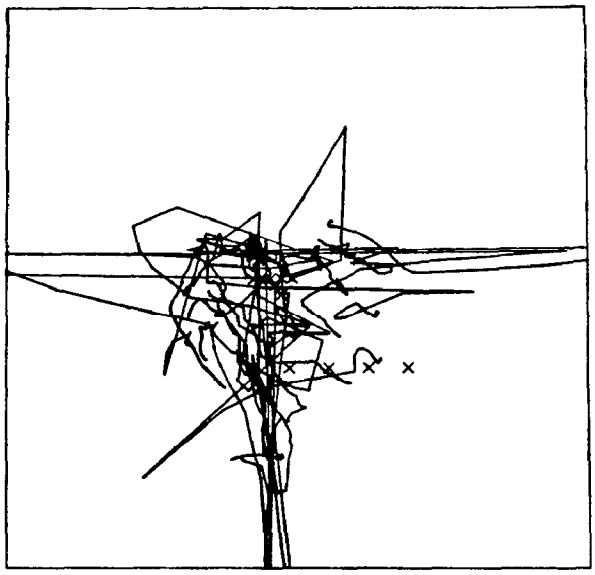
A literature search did not reveal any prior use of heuristic methods in the NR solution of the FDP problem. The method used here is kindred to acceleration methods such as Steffensen iteration [3] in that it uses estimates of the next solution interleaved with more precise iterations. Steffensen iteration could not be directly applied in the present case, however, because using both it and NR is redundant. Both are extrapolation methods that use the iteration function in a precise manner [3]. The heuristic extrapolation used here is more adaptable. It does not require a function evaluation, and makes use of properties of the solution specific to this problem domain.

Problem Formulation

The formulation of FDP given in [1] is used to test the validity of the methods. Solution is in two phases. The first uses only an attraction force in order to quickly establish relative positions from a random initial placement. The second phase adds a repulsion force to obtain the final solution. There are two potential difficulties in this formulation, both involving the repulsion force.

With the repulsion constant defined as in [1], there is an implicit mismatch between the strengths of the attraction force, which is tied to distance units, and the repulsion force, which is not. This means the relative strengths of the attraction and repulsion forces, and hence the degree of device spreading, will depend on the coordinate system used, which is undesirable [4]. A normalization is needed.

A second problem is encountered in the partial derivative with respect to distance of the repulsion force. Although the repulsion used in [1] is independent of distance, the relative strengths of the X and Y components change with the angular orientation of device pairs. The rate of change is highest when the devices are close together, and is of the opposite sense to the partial derivative of the attraction force. If many devices are close together, the addition of many partials of comparable magnitude and opposite sign can result in a very small quantity for the total partial derivative of force, F′. As the expression for device movement in the NR method is F/F′, this results in an anomalously large movement of the device. This shows as devices being (temporarily) ejected from the group, mostly along the coordinate axes (Figure 2). This effect is particularly noticeable at the beginning of the second phase, when a cluster of devices exists from the attraction-only solution in the first phase.



x - initial position

Figure 2. Instability due to repulsion derivative

The FDP model has been found to have desirable properties [1][2]. Its metric corresponds directly to a relevant physical quantity--connection length. It does not depend on a placement grid, and is therefore suited to placement of mixed sizes of devices. The repulsion force can be used to reduce device congestion and reduce or prevent device overlap. The NR solution procedure applied to this problem has been characterized as well-conditioned, stable, and finding a true minimum rather than a local one [1][2]. Because of these attributes, it is actively used in industry as well as being of interest theoretically.

The Heuristic Acceleration Methods

Any reduction in the calculation required for an iteration or reduction in the number of iterations will yield a corresponding increase in efficiency. The first method described below decreases the total number of iterations. The second does both; faster heuristic iterations are introduced to take the place of one or more non-heuristic iterations. In the following, a non-heuristic iteration will be called a step, and a heuristic iteration called a jump.

The instability caused by the repulsion term can be greatly reduced by use of knowledge of the solution state. Although the devices are close together after the first phase, they are going to be approximately equally distributed after the second phase due to repulsion. In a lattice of devices spaced at a distance of D units, the maximum partial derivative of repulsion due to any given device is 1 (for the device immediately above or to the side). If the value of the partial derivative of repulsion due to any device is limited (clipped) to 1 it will reduce the instability in the early part of the second phase. In the later part of the second phase, most of the values are less than 1 anyway, so the limited value can be regarded as a good approximation.

Note that this limiting does not change the calculation of forces (the iteration function), only of displacements. Given that the final solution is insensitive to initial device position [1][2], one is assured that this procedure is safe. That is, if it converges, it is approaching the same solution given by the more exact (and unstable) formulation.

The second method is motivated by observing two characteristics of iterative solutions using the FDP model. First, the successive positions of devices during iterative solution tend to define smooth paths (Figure 1). Second, instability is manifested by oscillation, as a result of over-shooting. These two properties are used to define a heuristic for extrapolation. Since the extrapolation calculation involves only a few points rather than all the other devices, a jump takes less computation than a normal iteration.

Observation of mis-extrapolated device paths indicates instability is manifested in a zig-zag pattern. This corresponds to the overshoot and oscillation one expects intuitively in a system of springs. In the context of steps and jumps, if a jump mis-extrapolates (zig), the subsequent steps will try to bring it back (zag). For this reason, the extrapolation method should resist rapid changes in path curvature.

The extrapolation is utilized by taking enough steps to define a path, then taking a jump based on an extrapolation of that path. More steps are taken to stabilize the path, followed by another jump, and so on. The parameters in this alternation of steps and jumps are the number of successive steps and the number of steps ahead the jump is predicting (jump size).

The need for stability is easily demonstrated by the naive approach to acceleration. This is done by increasing the constant that determines the amount of device movement in the NR iteration. This constant is .5 in [1]; a few experiments at values above .8 suffice to show that wildly unstable and frequently divergent behavior results.

A bad estimate of position may result in the post-jump position being much different from the one that would have been reached by steps. This will disturb neighboring devices and delay the solution process. There are a number of possibilities for reducing this effect. Most simply, the jump size can be limited to a small number. This limits the propagation of error due to inaccurate extrapolation. One could also save the old positions of the devices, and backtrack if a jump does not decrease the net force.

The above suggestions do not make use of the underlying assumption that most extrapolations will be accurate. A more pro-active treatment can be imagined in a situation where the number of devices that has been mis-extrapolated is relatively small. A jump could be followed with a step in which only the positions of the mis-extrapolated devices are changed in order to bring them into relative equilibrium with their more well-adjusted fellows. These devices are identifiable by their relatively large calculated displacements. The calculation required is still that of a normal step, but the need for additional iterations to smooth the disturbances of neighbors is reduced.

Implementation of Heuristic Acceleration

In [1] there are two parameters that were chosen empirically, the constant of repulsion (Cr) and the termination criterion (E). Prediction of their best value was noted as an open problem, which this paper does not address. To provide a meaningful basis for comparison, these parameters are fixed for all tests below. The methods presented here are insensitive to variations in these parameters. Cr is .33 for boards IL25 and RL25, and .16 for IL100 and RL100. These values are chosen to spread components evenly over the board area. E is 1 for IL25 and RL25, and 30

for IL100 and RL100. These values correspond to the cessation of significant relative movement of devices.

The partial derivative of repulsion for any device is limited to the maximum expected equilibrium value of 1, as mentioned above. In addition, the partial derivative of the repulsion term is omitted in the calculation for the first four iterations of the second pass, when the devices are close together. Data on the effect of this choice of values appears below.

The extrapolation used below takes the last two movements of the device, considered as vectors, and calculates the angle between them and the ratio of their lengths. This angle is used in extrapolating, but to reduce zig-zag instability, the angle is limited to a maximum value. Again using the smooth path heuristic, this value is chosen to be a small angle sufficient to follow the general curvature of the paths. This value is not critical; a value of .1 radians is used here. A jump of size S is taken by successively adding S vectors with this same angular change and length ratio.

Experimental Method and Results

The solution method given in [1] was implemented along with the improvement suggested there of allowing repulsion only between connected devices. This implementation was tested, without heuristic acceleration, on a number of boards for which results were given in [1]. These same boards were then used for the heuristic acceleration experiments. The boards are designated RL25, RL100, IL25, and IL100 (these are regular--RL--and irregular--IL--lattice designs, where the imbedded number is the number of movable devices).

In the tables presented below, computation time is given in CPU seconds. Implementation was in the C language with an optimizing compiler. All tests were done on an HP 9000 Series 320 technical workstation running HP-UX. HP-UX is derived from Unix (Unix is a trademark of AT+T) and adheres to AT+T's System V interface definition Issue 1.

Table 1 gives the number of steps in the second (repulsion) phase with and without the heuristic limit on the partial derivative of repulsion. There is an average two-fold speed increase with the limit. As well as reducing the expected number of iterations, this stabilization also has the effect of further smoothing the device paths, which increases the efficiency of the extrapolation. In the experiments below, the repulsion derivative is always limited.

Board:	RL25	IL25	RL100	IL100
Limited dR:	64 it.	82 it.	45 it.	34 it.
Unlimited dR:	57 it.	273 it.	61 it.	85 it.
Speed gain:	.89x	3.32x	1.35x	2.5x

Table 1. Effect of Limiting Repulsion on Convergence

Experimentation on the extrapolation method is structured by varying two parameters: the number of steps between jumps and the jump size. The overall effect on computation time for both phases combined is shown in Table 2. Rows correspond to the number of steps between jumps. Columns correspond to the jump size, and table values are in CPU seconds as noted above. A dash for steps and jump size indicates no heuristic acceleration. The value in the dashed row and column (at the upper left of each sub-table) is the one to which the others are compared to evaluate the effectiveness of the method.

Jump Size

Number of steps between jumps

	-	1	2	3	4	5	6	7
-	18.4	-	-	-	-	-	-	-
2	-	12.2	12.9	11.2	9.8	10.4	7.7	11.1
3	-	18.4	14.5	11.6	11.8	11.1	11.1	12.6
4	-	11.6	12.0	10.2	8.0	9.1	10.6	11.8
5	-	15.5	15.7	14.3	15.3	10.8	12.5	12.9

Board RL25

	-	1	2	3	4	5	6	7
-	162.9	-	-	-	-	-	-	-
2	-	154.9	186.7	212.7	189.0	139.9	131.3	99.7
3	-	281.5	297.1	248.2	201.9	162.8	108.5	130.2
4	-	199.7	137.4	198.7	172.3	89.7	86.8	126.3
5	-	248.4	143.5	201.4	194.4	230.4	90.4	140.5

Board RL100

	-	1	2	3	4	5	6	7
-	23.1	-	-	-	-	-	-	-
2	-	16.6	13.1	12.3	11.8	11.4	10.5	7.2
3	-	18.0	14.8	13.2	12.4	11.6	11.7	10.8
4	-	18.9	16.6	14.5	13.4	12.3	11.5	11.1
5	-	19.5	17.2	15.7	14.5	13.5	12.5	9.1

Board IL25

	-	1	2	3	4	5	6	7
-	114.5	-	-	-	-	-	-	-
2	-	93.4	67.4	72.3	61.2	56.1	59.4	59.2
3	-	105.9	78.5	86.1	81.0	61.8	61.7	55.6
4	-	103.9	97.8	76.8	86.0	114.8	79.1	77.7
5	-	104.4	102.9	88.7	108.9	75.7	65.4	67.4

Board IL100

Table 2. Run Times for Variations in Step/Jump Parameters (in CPU seconds)

Jump Size

Number of steps between jumps

	-	1	2	3	4	5	6	7
-	1.0	-	-	-	-	-	-	-
2	-	.51	.33	.24	.23	.25	.24	.27
3	-	.37	.29	.29	.29	.31	.40	.32
4	-	.56	.65	.41	.41	.65	.60	.41
5	-	.50	.70	.46	.42	.33	.66	.43

Board RL100

	-	1	2	3	4	5	6	7
-	1.0	-	-	-	-	-	-	-
2	-	.73	.55	.50	.47	.39	.43	.41
3	-	.61	.66	.63	.55	.51	.45	.44
4	-	1.01	.86	.84	.92	.73	.59	.53
5	-	.88	.75	.71	.59	.72	.59	.72

Board RL25

	-	1	2	3	4	5	6	7
-	1.0	-	-	-	-	-	-	-
2	-	.59	.61	.48	.45	.41	.35	.31
3	-	.65	.67	.56	.50	.55	.47	.46
4	-	.76	.69	.70	.57	.42	.50	.48
5	-	.82	.73	.72	.55	.66	.68	.61

Board IL100

	-	1	2	3	4	5	6	7
-	1.0	-	-	-	-	-	-	-
2	-	.65	.62	.54	.48	.43	.40	.36
3	-	.74	.72	.68	.64	.58	.51	.46
4	-	.78	.76	.74	.69	.68	.66	.63
5	-	.83	.80	.78	.76	.73	.70	.65

Board IL25

Table 3. Relative Accuracy of Extrapolation

Table 3's format is similar, except that instead of CPU seconds it gives the effectiveness of the heuristic jump. This is given by the ratio

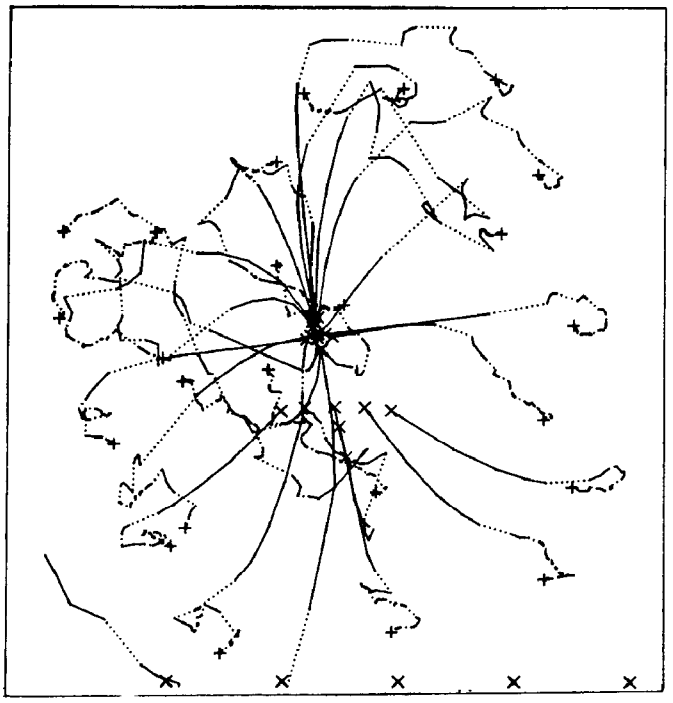
$$\frac{\text{(iterations in non-heuristic run) (steps)}}{\text{(iterations in heuristic run) (jump\_size + steps)}}$$

If the extrapolation was perfect, a jump of size N would replace N steps, and this number would be unity.

Analysis

One would expect that computation time would decrease with increasing jump size, until the effects of inaccurate extrapolation would become large enough to offset the gain. This is supported by Table 2, where run time generally decreases up to a step size of six. A path diagram for a jump size of six indicates that inaccuracies in the extrapolations are causing the paths to lose their smoothness, and device movement is wasted in regaining equilibrium positions (Figure 3).

The behavior shown in Figure 3 indicates that the extrapolation method used here is still prone to zig-zag instability, probably because it only uses the previous two points. Investigation is continuing on improved extrapolation methods that use more points to define a smoother extrapolation



x - initial position + - final position ... - jump

Figure 3. Corrections following large jumps

curve. This is expected to significantly increase the performance of the method.

Table 2 indicates that three or four steps between each jump generally gives best results. This is the length of the period of readjustment needed to establish a stable basis for extrapolation. On the other hand, there is no benefit in delaying after stability has been re-established, as the values for a five step interval indicate.

Table 3 shows the effectiveness of the jumps, using the ratio explained above. The effectiveness of the jumps corresponding to minimum CPU times varies between .35 and .6. As jump size increases, the effectiveness of the jump generally decreases. This is due to the propagation of error in extrapolation. After large jumps, additional steps are required to re-establish stability, partially negating the savings from the jump. Comparison between Tables 2 and 3 indicates that the minimum CPU time does not correspond with the most effective jumps. This illustrates a tradeoff where a degree of inaccuracy can be accepted because of the savings in time due to the faster extrapolation calculation.

The results show that a jump/step ratio of 6/4 provides an average 80% increase in speed. Together with the two-fold increase gained by stabilizing repulsion, this indicates that a practical implementation can get a three- to four-fold speed increase on a variety of boards with a single choice of parameters.

Conclusions

The heuristic methods presented above have been demonstrated to yield a three- to four-fold acceleration of the solution process. The methods are based on stabilization and extrapolation of device movement during iteration. The savings is especially valuable for large problems. The success of the extrapolation heuristic is due to the smooth paths described by device positions during iteration, the stability of the extrapolation, and the speed of the extrapolation relative to an iteration.

The method is currently limited by the inaccuracy in extrapolation of device positions. Incorporation of more accurate extrapolations or methods for selectively handling mis-extrapolated devices is expected to yield further improvements.

Although suggested by the FDP problem, the extrapolation method is generally applicable to iterative solution methods where convergence is slow, computation per iteration is high, and movement of successive solution estimates is by mostly smooth paths.

Opportunities for further development may exist in applying this method to other NR/FDP formulations. In particular, a partial plotting of device paths in [2] suggests that a similar approach may be effective.

Acknowledgements

I would like to thank Dr. N. Quinn for being kind enough to provide additional information on FDP in a number of telephone conversations. I am grateful to my project team, particularly Elaine Regelson and Chuck Fowler, for their support and encouragement of this research. I would also like to thank the reviewers for their helpful comments.

References

[1] Quinn, N. and Breuer, M., "A Force Directed Component Placement Procedure for Printed Circuit Boards", IEEE Trans. on Circuits and Systems, v. CAS-26, p. 377-388, June 1979.

[2] Antreich et al., A New Approach for Solving the Placement Problem Using Force Models, Proc. Int. Symp. on Circuits and Systems, IEEE, 1982.

[3] Conte, D. and de Boor, C., Elementary Numerical Analysis, pp. 95-109, 3rd ed., McGraw Hill, 1980.

[4] C. Fowler, personal communication.