

**Locating Input and Output Points in Facilities Design: A Comparison of Constructive,
Evolutionary and Exact Methods**

Rifat Aykut Arapoglu and Bryan A. Norman
Department of Industrial Engineering
University of Pittsburgh
1048 Benedum Hall
Pittsburgh, Pennsylvania 15261 USA
banorman@engrng.pitt.edu

Alice E. Smith, Senior IEEE Member
Department of Industrial and Systems Engineering
Auburn University
206 Dunstan Hall
Auburn University, Alabama 36849-5346 USA
aesmith@eng.auburn.edu

Abstract: This paper formulates and compares four new approaches to optimally locating the input and output station for each department within a facility design such that material handling costs are minimized. This problem is an NP-hard combinatorial problem with many real life applications of considerable economic consequence. A genetic algorithm (GA) is shown to be an effective and efficient optimization method when compared to integer programming, simulated annealing and three versions of a greedy constructive heuristic on a suite test problems of varying size. Seeding versus random initialization of GA populations are compared.

Revised for *IEEE Transactions on Evolutionary Computation*

July 1999

1. Introduction to the Problem

Facility design problems are a family of design problems involving the partitioning of a planar region into departments or work centers of given area, so as to minimize the costs associated with projected interactions between departments. These costs usually reflect material handling costs among departments based on the volume of material handling, distance to be transported and cost per unit distance. The material moves from the input/output point of a department to the input/output point of the next department on the routing specification. For a given facility, generally a subset of material will visit a department and the routings for each product or batch will be different. Such problems occur in many organizations, including manufacturing cell design [3], hospital design [10], land use planning [6] and construction site management [28]. For U.S. manufacturers, between 20% to 50% of total operating expenses are spent on material handling and an appropriate facilities design can reduce these costs by at least 10% to 30% [18]. Dr. James A. Tompkins, one of the seminal researchers in the field, recently wrote, "Since 1955, approximately 8 percent of the U.S. GNP has been spent annually on new facilities. In addition, existing facilities must be continually modified...These issues represent more than \$250 billion per year attributed to the design of facility systems, layouts, handling systems, and facilities locations..." [26]. Altering facility designs due to incorrect decisions, forecasts or assumptions usually involves considerable cost, time and disruption of activities. On the other hand, good designs can reap economic and operational benefits for a long time period. The problem primarily studied in the literature has been "block layout" which only specifies the placement of the departments, without regard for aisle structure, material handling system, machine placement within departments, or input/output stations (I/O) locations. Block layout is usually a precursor to these subsequent design steps, which are termed "detailed layout." Two recent survey articles

on the facility design problem are Kusiak and Heragu [17] and Meller and Gau [18].

Because of the computational complexities in optimizing multiple and non-linear objectives and constraints, only limited work has been done beyond block layout. The recent work of Benson and Foote [5] considers the placement of aisles and I/O locations after the relative location of the departments and the general aisle structure have been selected. Kim and Klein [16] consider the problem of placing I/O locations for a given automated guided vehicle network. They propose constructive heuristic methods for determining the I/O locations.

Additional related work on integrated facility design that considers machine placement within departments includes papers by Nagi and others [13, 15]. This work uses predefined departmental shapes set on a grid covering the facility space. In [13], Dijkstra's shortest path algorithm is used to calculate the rectilinear distance to and from pre-specified I/O locations. In [15], I/O points are placed during the optimization and a constraint is imposed to encourage aisles that are straight. Both papers use a simulated annealing heuristic to alter departmental placement. Another related work is by Banerjee et al. [3] where a genetic algorithm finds a "rough" design that is then fully defined using a subordinate mathematical programming routine. The number of I/O's per department is pre-specified and then they are optimally located with the department placement. The rectilinear distance is calculated between I/O points.

This paper focuses on the problem of optimal location of an I/O for each department given a specific block layout. The optimization approaches described are intended to work as subroutines for other optimization algorithms that search through block layouts, optimally locating the I/Os as a nested routine to evaluate the objective function (i.e., minimize material handling costs) of the total facility design. Because of the intended integrated nature of the optimization, computational effort versus accuracy becomes the most important point, an aspect

that has not been well studied by the papers cited above. Since the I/O optimization will be done many hundreds or thousands of times during facility design, computationally expensive approaches such as integer programming are prohibitive. Like the papers mentioned above, material flow both in and out of a given department is assumed to occur at the same physical point. However, the heuristic methods described in this paper can be straightforwardly generalized to separate the input point from the output point for specified departments. This will add constraints to the optimization problem and increase the search space; it is a subject of continuing research by the authors.

2. Problem Formulation

In this paper, it is assumed that a block layout has been constructed where all of the departments have rectangular shapes within a rectangular building (see Figure 1). The two most general mechanisms in the literature for constructing such layouts are the flexible bay [22, 25] and the slicing tree [8, 9, 12, 14, 23, 24]. However, these structures are not necessary for the approaches described in this paper. Subsets, such as the popular quadratic assignment problem (QAP) approach to block layout, can be handled in the same manner.

In this paper, *contour distance*¹, is being used [7, 19, 20, 21]. Traditionally, rectilinear and Euclidean distance measures have been used in layout design problems. Neither of these measures is capable of depicting the real distances experienced by material movement in a facility as they assume centroid to centroid rectilinear / Euclidean distances and permit material movement through other departments (see Figure 2). The contour distance metric, on the other hand, only permits travel along the boundaries of the departments and follows the shortest path connecting a given pair of departments from and to the I/O points, as shown in Figure 1.

¹ Also called “free flow” in some papers.

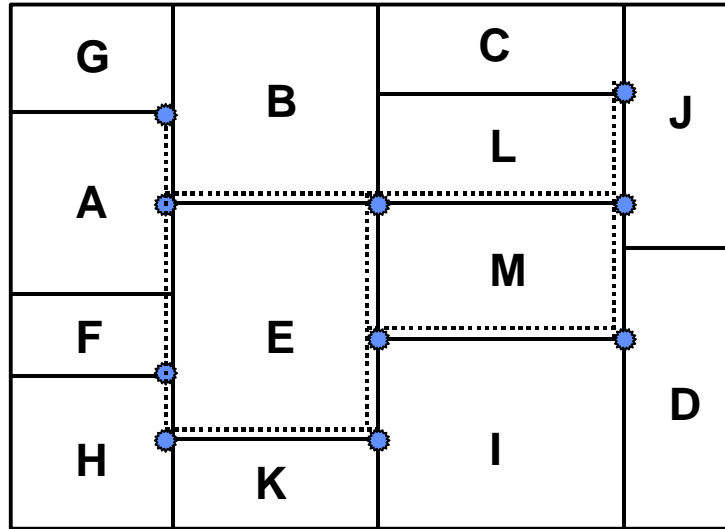


Figure 1. Flexible bay design with 13 departments, 10 I/O locations and material travel along department perimeters (dashed lines).

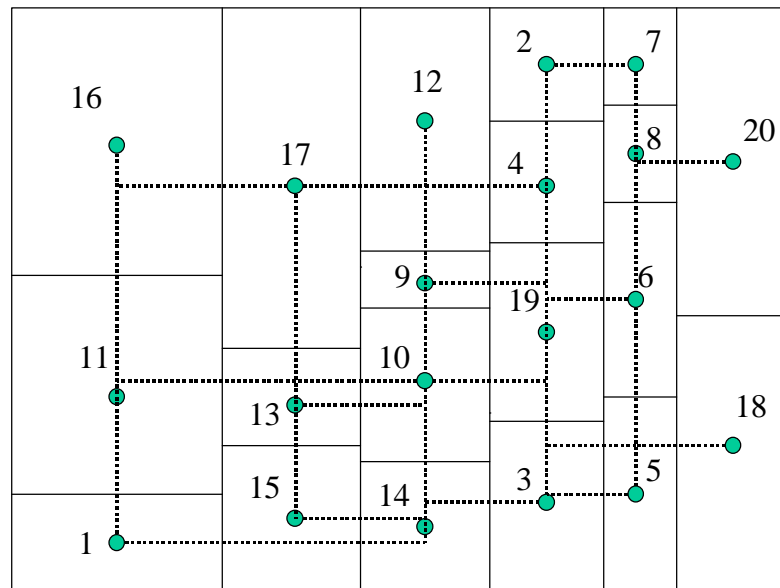


Figure 2. Block layout design of 20 departments showing the impracticalities of using the rectilinear distance metric between departmental centroids.

The single input/output (I/O) point placement problem, i.e. one I/O per department, in facilities design can be described as a minimization problem that uses the notation below.

Notation:

$X_{i,k}$ = binary variable that equals 1 if department i uses I/O candidate location k , zero otherwise

$X_{i,k,j,l}$ = binary variable that equals 1 if department i uses I/O candidate location k and department j uses I/O candidate location l , zero otherwise. It indicates which path is used for moving material from department i to department j .

$d_{i,k,j,l}$ = the distance from candidate I/O location k in department i to candidate I/O location l in department j

$d_cost_{i,j}$ = distance from department i to department j using the I/O that is assigned for each department

Loc_i = the set of candidate I/O locations for department i

$f_{i,j}$ = the flow (material quantity times cost per unit distance) between departments i and j

N = the number of departments in the layout

The objective of the optimization problem is to minimize the product of flow times distance for the layout. The flow amount is fixed but the flow distances traveled between the departments depend on where the I/O locations are placed in each department. The I/O location problem can be represented as the following integer program.

$$\text{Min } \sum_{i=1}^N \sum_{j=1}^N f_{ij} d_cost_{ij} \quad (\text{OF})$$

Subject to:

An assignment constraint that requires each department to have exactly one I/O location:

$$\sum_{k \in Loc_i} x_{ik} = 1 \quad \forall i = 1, \dots, N \quad (\text{C1})$$

Cost calculation constraints:

$$\sum_k \sum_l x_{i,k,j,l} = 1 \quad \forall i = 1, \dots, N, j = 1, \dots, N \quad (C2)$$

$$x_{i,k,j,l} \leq x_{i,k} \quad \forall i = 1, \dots, N, j = 1, \dots, N, k \in Loc_i, l \in Loc_j \quad (C3)$$

$$\sum_k \sum_l x_{i,k,j,l} d_{i,k,j,l} = d_{cost_{i,j}} \quad \forall i = 1, \dots, N, j = 1, \dots, N \quad (C4)$$

Constraint C2 insures that a path is selected for moving material from department i to department j . Otherwise the objective function would force $x_{i,k,j,l}$ to be zero for all k, l pairs where $k \in Loc_i$ and $l \in Loc_j$, no path would be selected and the distance would be zero. Constraint C3 permits $x_{i,k,j,l}$ to equal one only if both $x_{i,k}$ and $x_{j,l}$ equal one (the material flow path can only go from I/O location k of department i to I/O location l of department j if k and l were chosen as the I/O locations of departments i and j , respectively). Constraint C4 evaluates the distance of the path from department i to department j . Note that Constraints C2 to C4 could be replaced with the single alternative constraint, CA, given below but this Constraint is non-linear and integer which would make the resulting math program extremely difficult to solve.

$$\sum_k \sum_l x_{i,k} x_{j,l} d_{i,k,j,l} = d_{cost_{i,j}} \quad \forall i = 1, \dots, N, j = 1, \dots, N \quad (CA)$$

3. Complexity of the Single I/O Placement Problem

The following two observations from [20] make possible the design and analysis of the single I/O point location problem:

- The total number of potential I/O points can be reduced to the finite set of departmental intersection points. Consider the example in Figure 3. The candidate I/O locations for department B are locations 1, 2, 3, 8, 9, and 10.²

² While the optimal location of I/O points will always be at departmental intersections, it may not be physically practical, in some cases. A perturbation along the relevant department perimeter near the optimal intersection will maintain near optimality of the cost objective while being physically tractable.

- There can be at most $2N-2$ such points for a flexible bay block layout.

Note that more than one department may share the same I/O location as in departments C and J of Figure 1. Since each department may use only one of its I/O points independently of the other departments, the total number of possible combinations grows exponentially, creating an NP-hard problem [11]. Namely,

$$\prod_{i=1}^N |\text{Loc}_i|$$

For the layout in Figure 3, this quantity is 2.9×10^7 . This number is on the order of 10^{13} for a facility with 20 departments.

4. Optimization Methodologies

It is assumed that a block layout design with all rectangular departments located within a rectangular floorplan has already been identified and the optimization task at hand is to choose a single I/O location for each department such that material handling costs are minimized using the contour distance metric. The shortest paths between all pairs of candidate I/O points, which are located at the intersection points of departments (see Figure 3), need to be found. This is accomplished by creating a network with non-negative arc lengths that represent the distances between the I/O points using the perimeter distance metric. The nodes of the network include all of the potential I/O locations, which are found at the intersection points of the departments, and a node for each one of the departments. The Floyd-Warshall algorithm [1] is used to find the shortest distances between all pairs. This algorithm is polynomial and runs in $O(n^3)$ time where n is the number of network nodes.

To locate the optimum I/O point for each department, four approaches were developed and compared. An exact approach to solve this problem is through the use of integer programming (IP). This always finds the optimum solution, but the search space grows

exponentially with the number of candidate I/O locations, therefore precluding its use for medium to large sized facilities. A computationally expedient approach is a constructive heuristic that works in a greedy, or myopic, fashion. Three versions of the constructive heuristic were developed and tested. A third alternative is a simulated annealing heuristic using a neighborhood structure to evaluate moves. The fourth alternative uses genetic algorithms (GA) to improve upon a set of feasible, but randomly chosen, I/O locations. A lower bound was also devised by considering the linear relaxation of the mixed integer program.

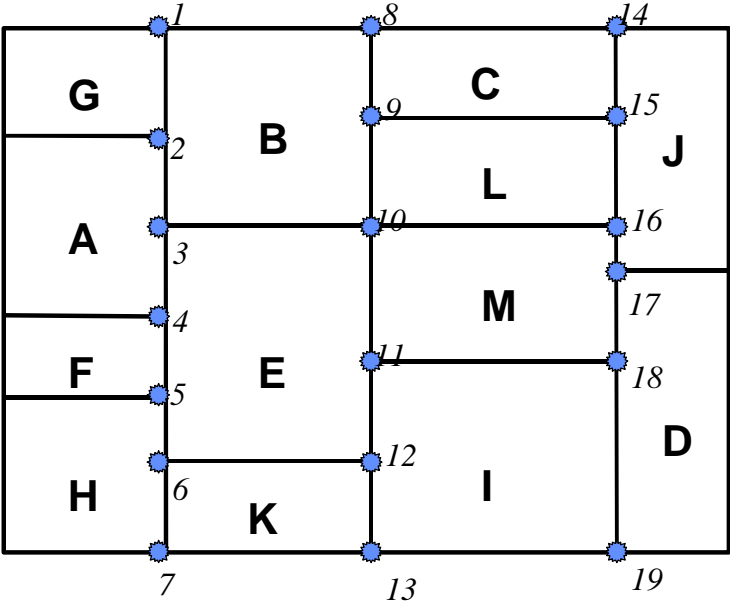


Figure 3. Candidate I/O locations for a flexible bay layout.

4.1 The Integer Programming Approach

The formulation developed in section 2 was coded as an IP and solved using CPLEX.

4.2 The Constructive Heuristic

Another approach is a very quick constructive heuristic that operates in a greedy fashion. There are three versions of this heuristic. Version A is a deterministic constructive version, version B is the same constructive heuristic but is not deterministic and version C uses the solution from

version A as the initial point for perturbations (i.e., version C is an improvement heuristic that is seeded with version A).

Version A is deterministic and assigns the I/O points in order of the magnitude of the material flows. The pseudocode is below.

Additional Notation:

A set of assigned departments

$S = \{1, 2, 3, \dots, N\}$ set of all departments

$totf_{i,k}$ total material flow using I/O point k of department i

Algorithm:

Step 1:

$$A = \emptyset$$

Set $totf_{i,k} = 0$ for all $i \in S$ and $k \in \text{Loc}_i$

Step 2:

Let $totf_{r,s} = \text{Max} \{totf_{i,k} : i \in S \setminus A, k \in \text{Loc}_i\}$

$A \leftarrow A \cup \{r\}$. Assign I/O point s to department r .

If $A = S$ then stop, all departments have been assigned an I/O point.

Else

$totf_{i,k} = 0$ for all $i \in S \setminus A$ and $k \in \text{Loc}_i$

Using the Floyd-Warshall algorithm, find the shortest paths between all node pairs in the network consisting of all candidate I/O locations for all departments in $S \setminus A$ and the selected I/O locations in all departments in A .

For each flow, f_{ij} that uses I/O point k of department i and I/O point l of department j , update

$$totf_{i,k} = totf_{i,k} + f_{i,j} \text{ and } totf_{j,l} = totf_{j,l} + f_{i,j}$$

Version B is identical to version A except for the first item of Step 2. Rather than being a deterministic greedy selection (Max), the selection is made according to a probability. First, the maximum total flow I/O point for each department is identified. These are sorted according to flow, and those below the mean are deleted. Probabilities are assigned to the remaining maximum flow I/Os, and the next I/O assigned is selected based on a uniform random number and these probabilities.

Version C takes the solution from version A and perturbs the I/O locations, department by department, for each department. The best perturbation is taken as the move. The procedure stops when there is no improving move possible. While slower than either version A or B (which are fully constructive), this improvement version is nonetheless very speedy.

4.3. Simulated Annealing

As a baseline for the GA described in the next section, a simulated annealing (SA) approach was coded. Starting with a random solution, a move to a neighboring solution is considered. The move space consists of moving one I/O point from its location to another feasible location. Improving moves are always accepted while non-improving moves are accepted with:

$$\Pr(\text{accept}) = e^{\left(\frac{-\Delta OF}{T}\right)}$$

where T is the temperature. The temperature changes with a geometric cooling schedule:

$$T_{new} = T_{old} * 0.90$$

with an initial temperature of 100 and a final temperature of 10. The number of moves per temperature is constant during the search and ranges from 1000 to 1400, depending on problem size. The SA was designed to run almost the same CPU time as the GA so a direct comparison could be made.

4.4. Genetic Algorithm

The overall flowchart of the genetic algorithm is given in Appendix I. The next sections discuss the particulars of the method.

Encoding and Fitness

A permutation encoding is used to represent a given allocation of I/O locations to the departments. Each candidate I/O location is numbered (to a maximum of $2N-2$) and it is possible that two neighboring departments may use the same *location* as their I/O. In that case the same I/O number is used to represent both I/Os. The example below denotes the encoding corresponding to the layout given in Figure 4 using the I/O numbering of Figure 3.

Department	A	B	C	D	E	F	G	H	I	J	K	L	M
I/O Locations	3	10	15	18	11	5	2	6	12	15	12	16	10

The fitness of a given solution is the cost of the material flow (i.e., equation OF) between each pair of departments using the selected I/O locations and the contour distance metric.

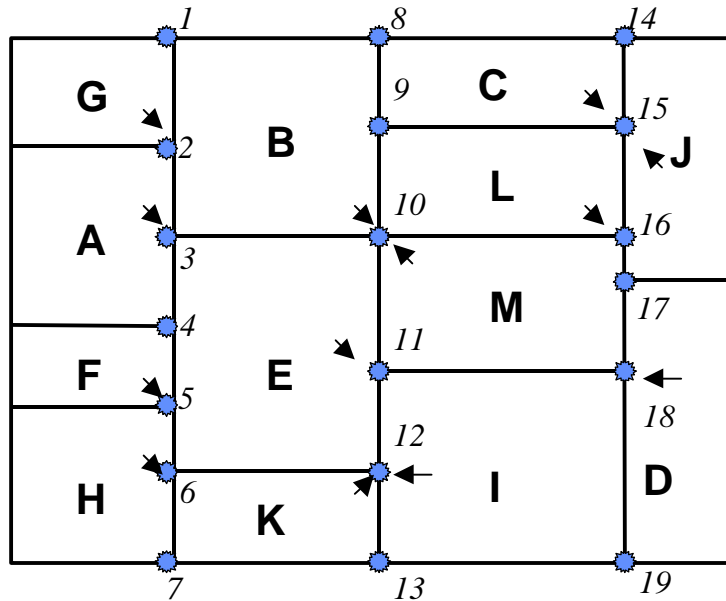


Figure 4. I/O locations for an example GA chromosome.

Selection, Crossover, Mutation

An elite set of 10 from a population of 50 is copied directly to the list of candidates to survive to the next generation. From the old generation of 50, one parent is selected using a tournament of size 2 while the other parent is selected randomly from the population (excluding the parent already selected via the tournament). Uniform crossover with probability of 1.00 is applied to produce one offspring, with a preference of 0.70 for alleles from the more fit parent. Note that every crossover produces a feasible allocation of I/O locations and thus, there is no repair needed. This continues until 50 offspring are created each generation.

The offspring are subject to mutation with a probability of 0.5. Every allele is tested for mutation individually and independently with a preselected mutation probability of 0.10. If an allele is chosen for mutation, the new allele is selected randomly from the feasible I/O locations for that department, not including the current value. Again, every mutation results in a feasible allocation of I/O locations. A wide variety of combinations of mutation probabilities were tried with the result that the GA was very robust from mutation probabilities ranging from 0.05 to 1.0, however 0.5 was marginally better than the others.

The 50 mutated offspring are combined with the ten elite solutions from the old population and sorted. The top 50 solutions are kept to form the new population.

Termination Criterion

The GA is terminated after 500 consecutive generations without any improvement in the best objective function value.

5. Test Problems and Results

The four approaches were applied to instances of well-known test problems with either slicing tree or flexible bay structures: a 10 department problem by Van Camp et al. [27], a 12

department problem by Bazaraa [4] that was encoded in the GA as a 16 department problem to handle empty space within the facility as was done by Tate and Smith [25], two instances of a 14 department problem by Bazaraa [4], four instances of a 20 department test problem of Armour and Buffa [2], and four instances of a 30 department problem from Tam [23, 24]. Three new test problems with 40, 50 and 60 departments, respectively, were developed by the authors, and the associated department areas and flows are available by email.

The relaxed MIP lower bound, the objective function results of the IP, the constructive heuristic versions and a summary of five runs of the GA without seeding are compared in Table 1. The simulated annealing results are not shown because the GA dominated the SA for best solution with equivalent CPU times. It can also be seen that versions A and B of the constructive heuristic were equivalent, except for problem Armour and Buffa - 2. Version C improves on versions A and B every time with improvements ranging from 0.3% to 18% and an average improvement of 7%. The GA is very consistent with at least one seed finding the optimal solution for each problem instance with a known optimal solution except for the Armour and Buffa - 1. The GA is also consistent across different random number seeds with almost no variation across all of the test problems.

The best solution of the GA equals or betters the constructive heuristics in every case. Compared to versions A and B, the GA obtained improvements of 0.3% to 53%, with an average improvement of 13%. Compared to version C, the GA obtained improvements of 0% to 38%, with an average improvement of 6%. The magnitude of the improvement does not depend on problem size, rather it is the structure of the block layout and the material flow magnitudes and costs involved in each instance. The best I/O locations for each problem are given in Appendix II, as are the implied aisle structure, i.e. flowpaths.

Table 1. Computational Test Results.

Problem	Structure #	Lower Bound	Integer Program	CS – Version A	CS – Version B	CS – Version C	Best GA	GA CV
Van Camp et al. 10	FB	6214.94	7239.03	11118.31	11118.31	10007.85	7239.03	0.00
Bazaraa 12	FB	7116.79	8308.03	8692.03	8692.03	8308.03	8308.03	0.03
Bazaraa 14 –1	FB	2320.71	4223.14	5648.36	5648.36	4793.56	4223.14	0.00
Bazaraa 14 - 2	FB	4114.61	4991.78	5459.22	5459.22	5187.78	4991.78	0.02
Armour & Buffa 20 - 1	FB	223.17	344.82	388.59	388.59	357.02	345.89	0.01
Armour & Buffa 20 - 2	FB	202.19	334.94	371.20	364.43	345.68	334.94	0.00
Armour & Buffa 20 - 3	ST	391.39	461.42	545.28	545.28	497.32	461.42	0.00
Armour & Buffa 20 - 4	ST	315.47	441.87	549.56	549.56	490.45	441.87	0.00
Tam 30 -1	ST	12757.98	*	17100.97	17100.97	15314.14	15314.14	0.00
Tam 30 -2	ST	12846.20	*	16515.81	16515.81	15863.49	15853.01	0.00
Tam 30 -3	ST	2728.75	*	3819.43	3819.43	3459.11	3353.00	0.01
Tam 30 -4	ST	2920.04	*	4383.53	4383.53	4066.18	3746.07	0.01
Arapoglu et al. 40 - 1	ST	22409.94	*	24015.80	24015.80	23943.40	23942.40	0.00
Arapoglu et al. 40 - 2	ST	22524.86	*	25735.00	25735.00	25282.50	25282.50	0.00
Arapoglu et al. 50 - 1	FB	36453.30	*	40192.21	40192.21	39694.41	39694.41	0.00
Arapoglu et al. 50 - 2	FB	24327.50	*	41985.60	41985.60	40504.40	38333.81	0.01
Arapoglu et al. 60	FB	45071.36	*	51449.91	51449.91	50930.23	50930.23	0.00

FB (flexible bay) or ST (slicing tree)

* Cannot be solved optimally because of problem size.

A GA version that used one copy of the solution generated by the constructive heuristic, version A, as partial seeding for the initial generation was explored. The seeding strategy did not produce any conclusive results. For most runs, the best solutions generated by the seeded and non-seeded GAs were identical. In a few cases, the seeded solution was slightly worse and in a few cases, the seeded solution was slightly better. For this problem, using this version of seeding did not seem to impact the search process.

Table 2. Computational Time Results (Average CPU Seconds).

Problem	Structure #	CS – Version A	CS – Version B	CS – Version C	GA – 1 Run
Van Camp et al. 10	FB	0.01	0.01	0.02	2.78
Bazaraa 12	FB	0.04	0.05	0.08	5.38
Bazaraa 14 -1	FB	0.03	0.02	0.06	4.46
Bazaraa 14 - 2	FB	0.02	0.03	0.04	4.34
Armour & Buffa 20 - 1	FB	0.07	0.07	0.14	9.61
Armour & Buffa 20 - 2	FB	0.09	0.08	0.17	7.95
Armour & Buffa 20 - 3	ST	0.10	0.10	0.20	8.20
Armour & Buffa 20 - 4	ST	0.09	0.09	0.27	9.42
Tam 30 -1	ST	0.27	0.31	0.98	17.59
Tam 30 -2	ST	0.28	0.30	0.83	17.36
Tam 30 -3	ST	0.27	0.30	0.74	18.62
Tam 30 -4	ST	0.27	0.28	1.03	18.33
Arapoglu et al. 40 - 1	ST	0.67	0.66	1.25	32.32
Arapoglu et al. 40 - 2	ST	0.64	0.67	2.12	29.17
Arapoglu et al. 50 - 1	FB	0.98	0.99	2.53	51.65
Arapoglu et al. 50 - 2	FB	0.93	0.96	2.56	54.95
Arapoglu et al. 60	FB	1.83	1.90	4.94	84.44

FB (flexible bay) or ST (slicing tree)

* Cannot be solved optimally because of problem size.

A comparison of computational effort is germane. In Table 2, the CPU times in seconds on a Sun Ultra Enterprise-2 workstation with dual 200Mhz Sparc CPUs are presented for the constructive heuristic and one run of the GA. The IP, of course, took considerably more time for each problem instance, and could not finish for the larger (30 and more departments) problems. Versions A and B of the constructive heuristic are very quick. Version C is about 2 to 5 times longer than versions A or B, however there is a marked improvement in solution quality. For the

larger problems, the GA takes about 20 times longer than version C of the constructive heuristic. A strategy balancing computational effort with solution quality would use version C of the constructive heuristic as a subroutine optimization for most of the block layout design optimization and use the GA for subroutine optimization for the best few solutions of the population in the later generations.

6. Conclusions

The problem of locating I/O points for a block layout also implies the paths along which material will flow in the facility. Thus, it is the most important step of detailed facility design. To improve the optimal design of facilities, it is necessary to truly integrate the block layout of departments with choosing the I/O location for each department. The GA heuristic and the constructive heuristics devised in this paper make it possible to optimize the I/O locations (and thus the flowpaths) as a subroutine to block layout optimization. It is imperative that the optimization of the I/Os be computationally expedient so that this subroutine can be done for the many thousands of block layout candidates identified during the design phase. Because of the great computational effort of the integer programming method, it is only practical for very small problems, and the lower bound is too loose to be of much value during optimization. The constructive heuristics perform well and are very fast. The GA requires more computational effort but is able to match or improve upon their performance in every case with improvements ranging up to 53%. For the problems where the optimal solution could be determined using the IP the GA found solutions that were nearly always optimal. Where optimal was not found, the GA identified solutions that were *at most* 8.4% above the optimal on any given run.

Seeding the GA with the constructive heuristic solution did not significantly affect the performance although it is computationally trivial to do. For a complete optimization

methodology, where both block and detailed layout factors are considered simultaneously, it makes sense to use version C of the constructive heuristic for most of the GA search of the total design space and to use the I/O GA developed in this paper for only the better population members and / or at the end of the optimization procedure. The very best few solutions may be sent to the IP for the exact I/O optimization if the design problem is not too large. There are several extensions to this work. The first, as mentioned earlier, is to consider the input location separate from the output location for specified departments. The second is to consider intradepartmental material movement that may dictate certain departmental shape (handled by an aspect ratio or minimum side length constraint) or preferred I/O location. A preferred I/O location (for example, the I/O should be on the longer side) could be handled as a restriction (hard constraint), with a penalty (soft constraint) or even within the objective function (secondary objective). Both extensions can be handled within the same GA framework proposed although the constructive heuristic would need to be modified.

Acknowledgment

Alice E. Smith gratefully acknowledges the support of the U.S. National Science Foundation CAREER grant DMI 95-02134.

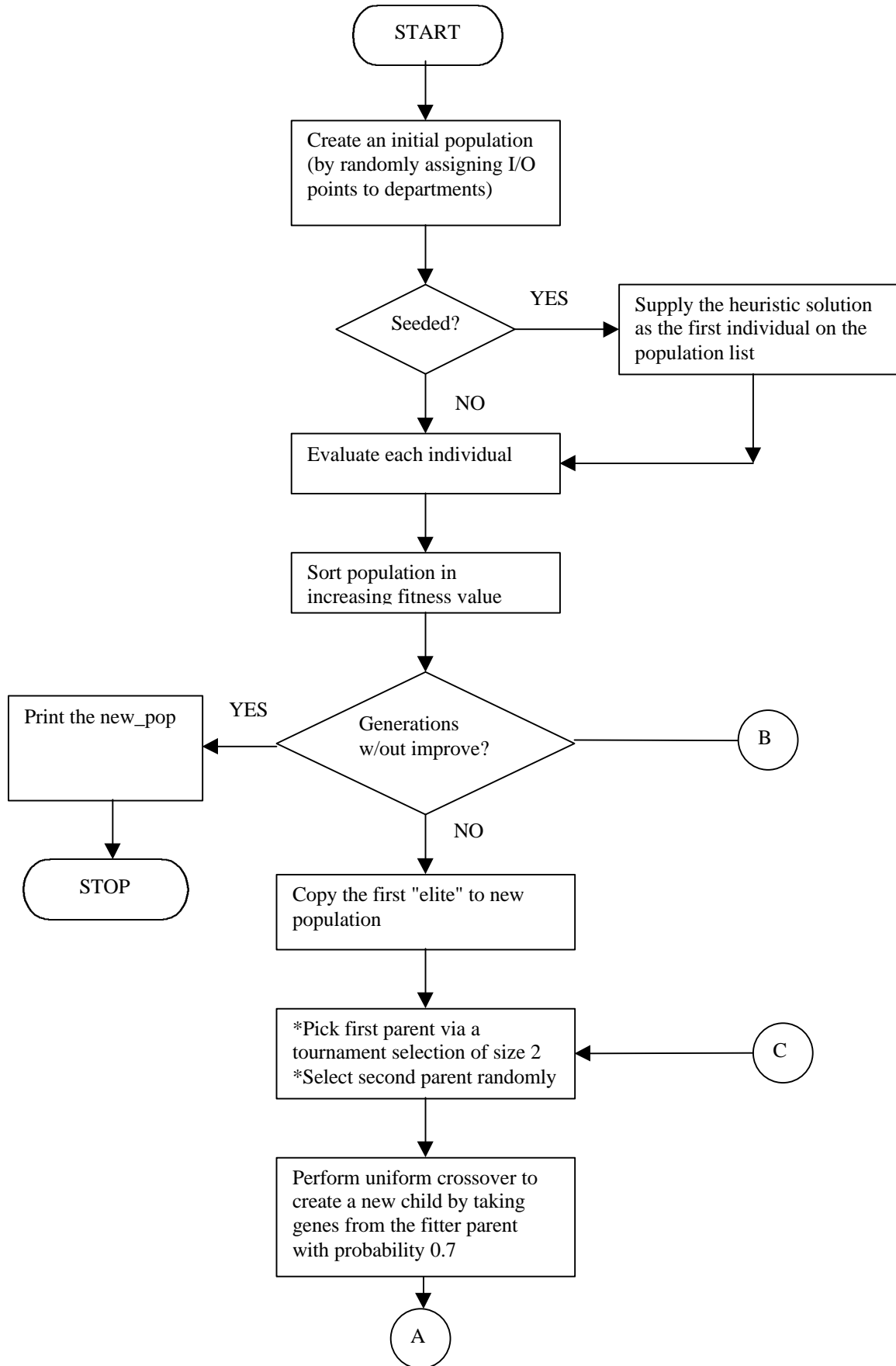
References

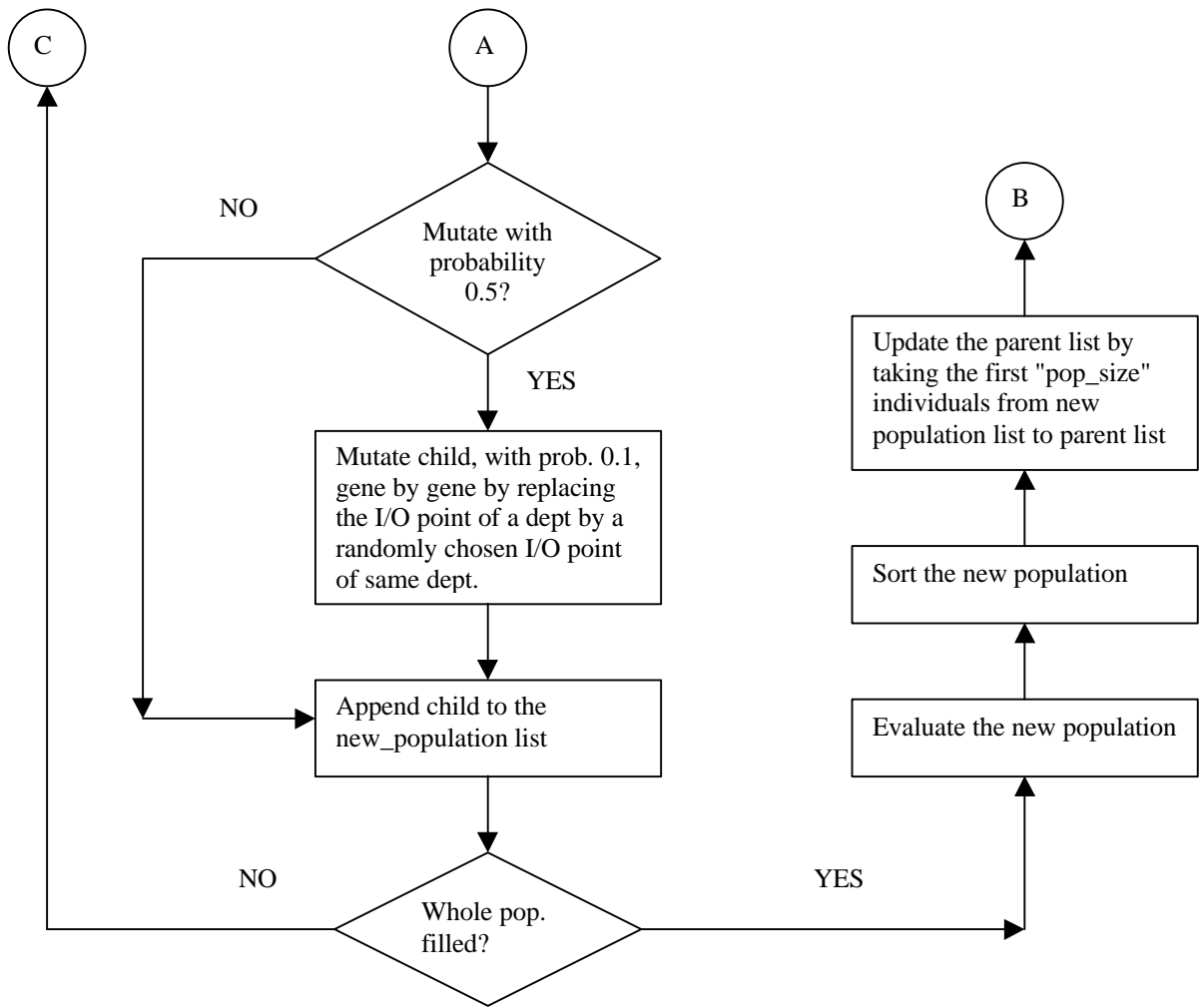
1. Ahuja, R. K., T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Upper Saddle, NJ, 1993.
2. Armour, G. C. and E. S. Buffa, "A heuristic algorithm and simulation approach to relative allocation of facilities," *Management Science*, 1963 **9**, 294-309.
3. Banerjee, P., Y. Zhou and B. Montreuil, "Genetically assisted optimization of cell layout and material flow path skeleton," *IIE Transactions*, 1997 **29**, 277-291.
4. Bazaraa, M. S., "Computerized layout design: a branch and bound approach," *AIIE Transactions*, 1975 **7**, 432-438.
5. Benson, B. and B. L. Foote, "Door FAST: a constructive procedure to optimally layout a facility including aisles and door locations based on an aisle flow distance metric,"

- International Journal of Production Research*, 1997 **35**, 1825-1842.
6. Bos, J., "Zoning in forest management: a quadratic assignment problem solved by simulated annealing," *Journal of Environmental Management*, 1993 **37**, 127-145.
 7. Chhajed, D., B. Montreuil and T. J. Lowe, "Flow network design for manufacturing systems layout," *European Journal of Operational Research*, 1992 **57**, 145-161.
 8. Cheng, R., M. Gen, and T. Tozawa, "Genetic search for facility layout design under interflows uncertainty," *Japanese Journal of Fuzzy Theory and Systems*, 1996 **8**, 267-281.
 9. Cohoon, J. P. and W. D. Paris, "Genetic placement," *IEEE Transactions on Computer-Aided Design*, 1987 **6**, 956-964.
 10. Elshafei, A. N., "Hospital layout as a quadratic assignment problem." *Operational Research Quarterly*, 1977 **28**, 167-179.
 11. Garey, M. R. and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York: W. H. Freeman and Company, 1979.
 12. Garces-Peres, Schoenfeld and Wainwright, "Solving facility layout problems using genetic programming," *Genetic Programming 1996 (Proceedings of the 1st Annual Conference – GP 96)*, MIT Press, 1996, 182-190.
 13. Harhalakis, G., T. Lu, I. Minis and R. Nagi, "A practical method for design of hybrid-type production facilities," *International Journal of Production Research*, 1996 **34**, 897-918.
 14. Kado, K., P. Ross and D. Corne, "A study of genetic algorithm hybrids for facility layout problems," *Proceedings of the Sixth International Conference on Genetic Algorithms*, Pittsburgh PA, July 1995, 498-505.
 15. Kane, M. C. and R. Nagi, "Integrated material flow and layout in facilities design," *Proceedings of the Sixth Industrial Engineering Research Conference*, Miami FL, May 1997, 919-924.
 16. Kim, J. and C. M. Klein, "Location of departmental pickup and delivery points for an AGV system," *International Journal of Production Research*, 1996 **34**, 407-420.
 17. Kusiak, A. and S. S. Heragu, "The facility layout problem," *European Journal of Operational Research*, 1987 **29**, 229-251.
 18. Meller, R. D. and K.-Y. Gau, "The facility layout problem: recent and emerging trends and perspectives," *Journal of Manufacturing Systems*, 1996 **15**, 351-366.
 19. Norman, B. A., A. E. Smith and R. A. Arapoglu, "Integrated facilities layout using a perimeter distance metric," *Parallel Problem Solving from Nature (PPSN V)* (A. E. Eiben, T. Baeck, M. Schoenauer and H.-P. Schwefel, editors), Lecture Notes in Computer Science 1498, Springer-Verlag, Berlin, Germany, 1998, 937-946.
 20. Norman, B. A., A. E. Smith and R. A. Arapoglu, "Integrated facilities layout using a contour distance measure," *IIE Transactions*, in revision.
 21. Sinriech, D., "Network design models for discrete material flow systems: A literature review," *International Journal of Advanced Manufacturing Technology*, 1995 **10**, 277-291.
 22. Smith, A. E. and D. M. Tate, "Genetic optimization using a penalty function," *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, 499-505.

23. Tam, K. Y., "Genetic algorithms, function optimization, and facility layout design," *European Journal of Operational Research*, 1992 **63**, 322-346.
24. Tam, K. Y., "A simulated annealing algorithm for allocating space to manufacturing cells," *International Journal of Production Research*, 1992 **30**, 63-87.
25. Tate, D. M. and A. E. Smith, "Unequal-area facility layout by genetic search," *IIE Transactions*, 1995 **27**, 465-472.
26. Tompkins, J. A., "Facilities planning: a vision for the 21st century," *IIE Solutions*, August 1997, 18-19.
27. van Camp, D. J., M. W. Carter and A. Vannelli, "A nonlinear optimization approach for solving facility layout problems," *European Journal of Operational Research*, 1991, 57, 174-189.
28. Yeh, I.-C., "Construction-site layout using annealed neural network," *Journal of Computing in Civil Engineering*, 1995 **9**, 201-208.

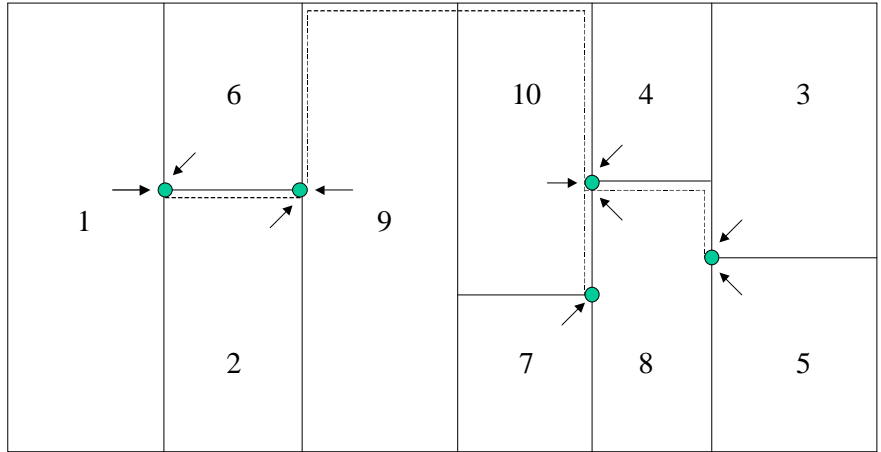
Appendix I, Genetic Algorithm Flowchart



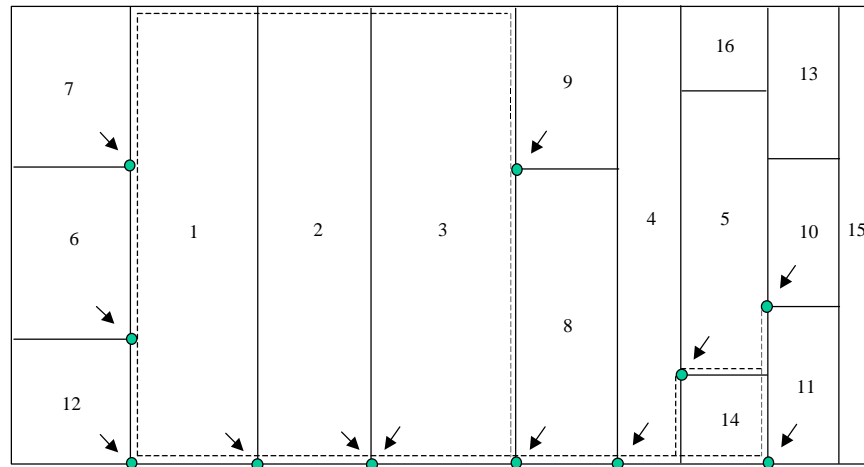


Appendix II, Block Layout, Best I/O Locations and Implied Flowpaths of Test Problems

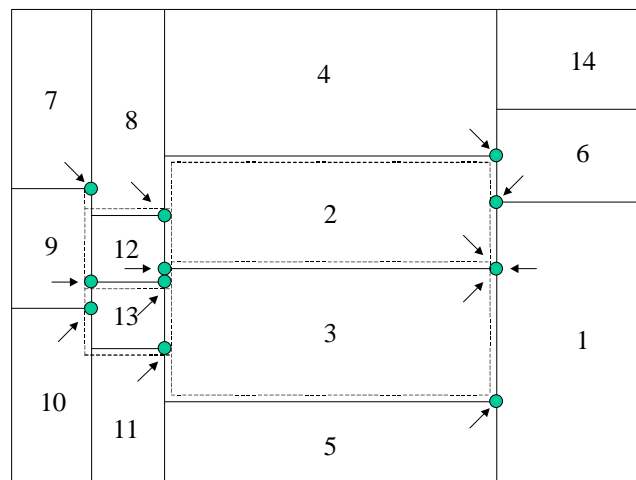
Van Camp et al. 10



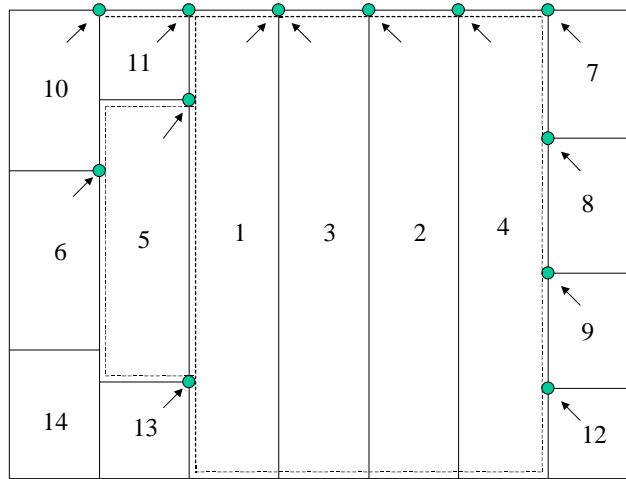
Bazaraa 12



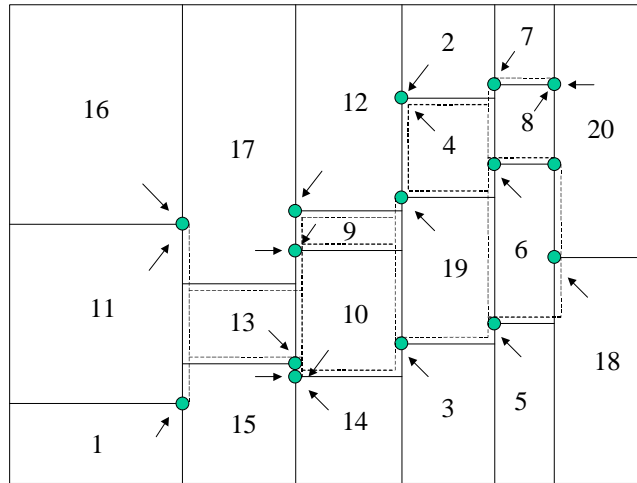
Bazaara 14 - 1



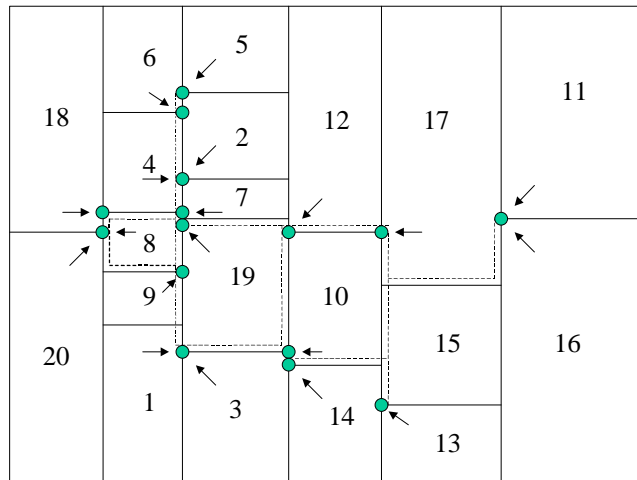
Bazaara 14 – 2



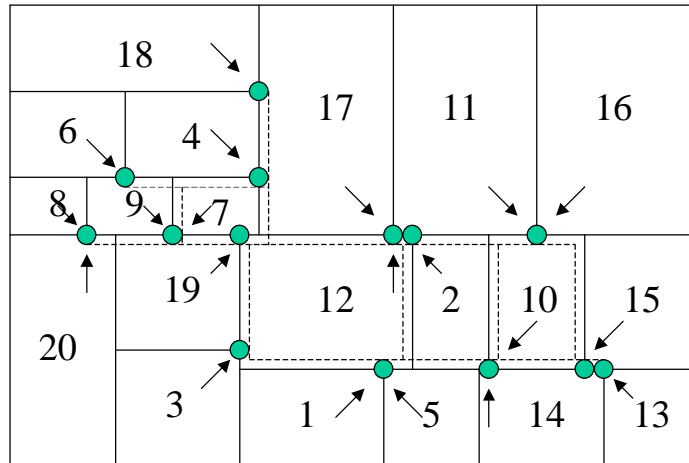
Armour and Buffa 20 – 1



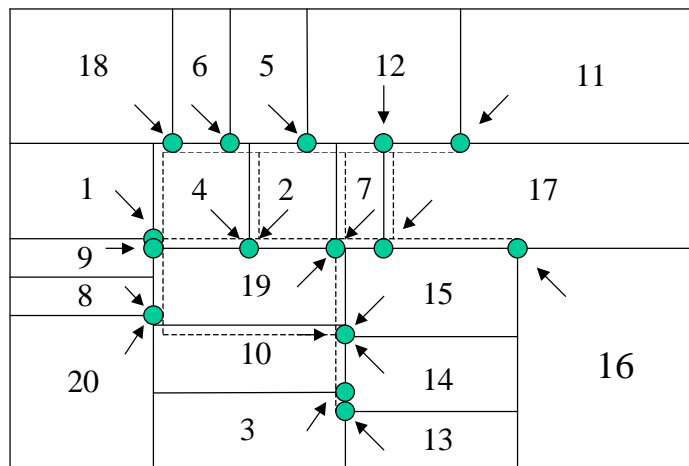
Armour and Buffa 20 – 2



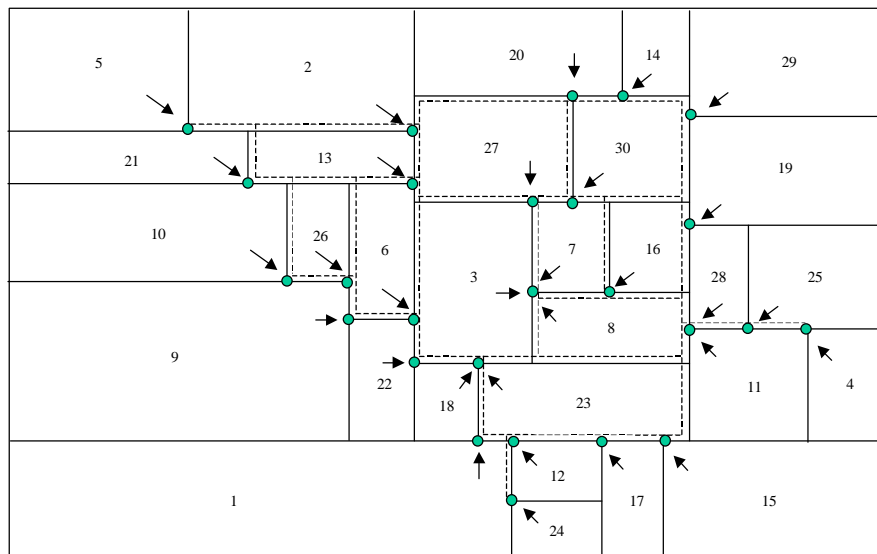
Armour and Buffa 20 – 3



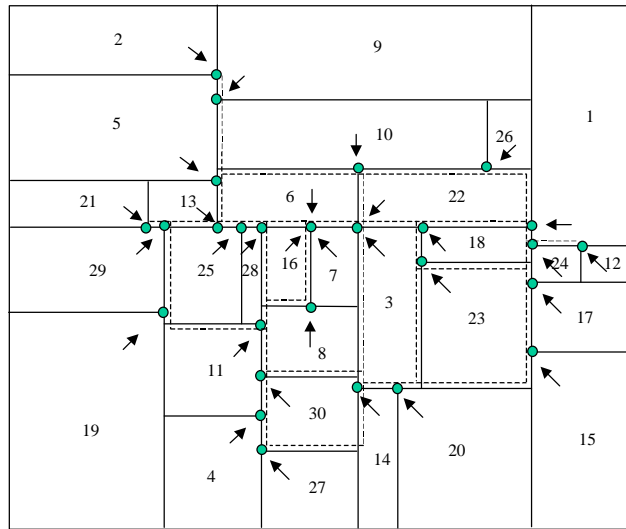
Armour and Buffa 20 – 4



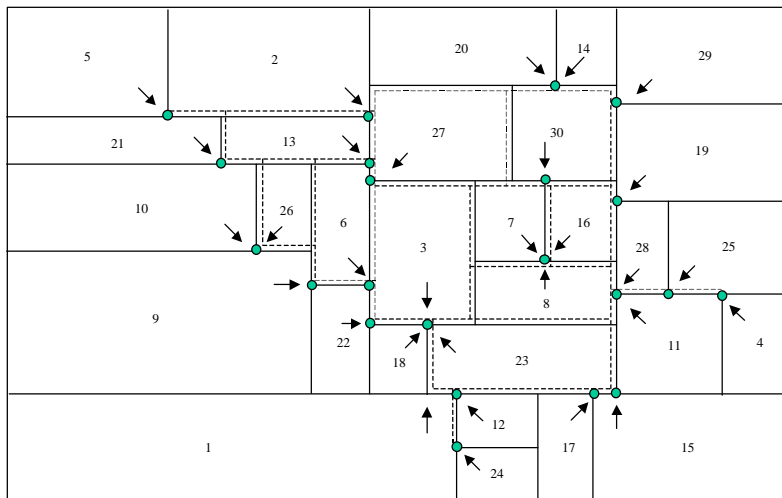
Tam 30 – 1



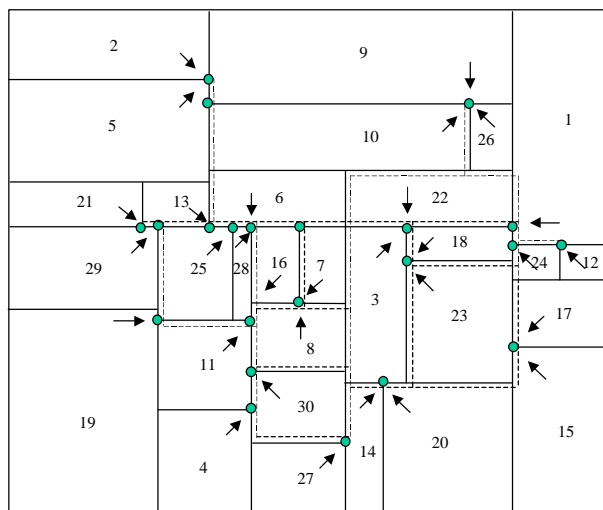
Tam 30 – 2



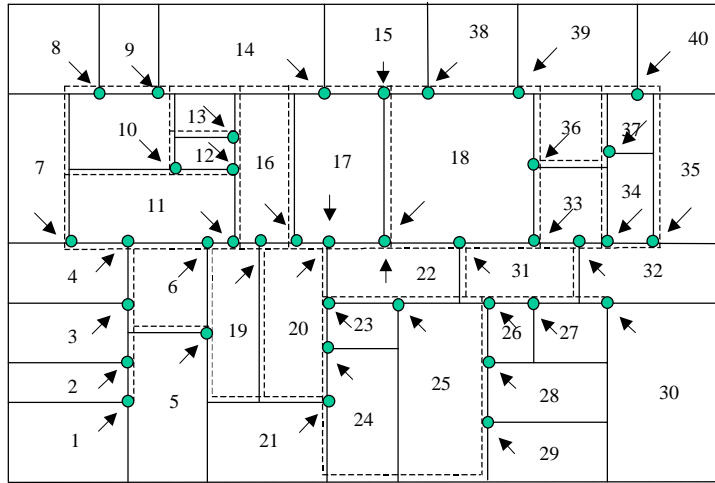
Tam 30 – 3



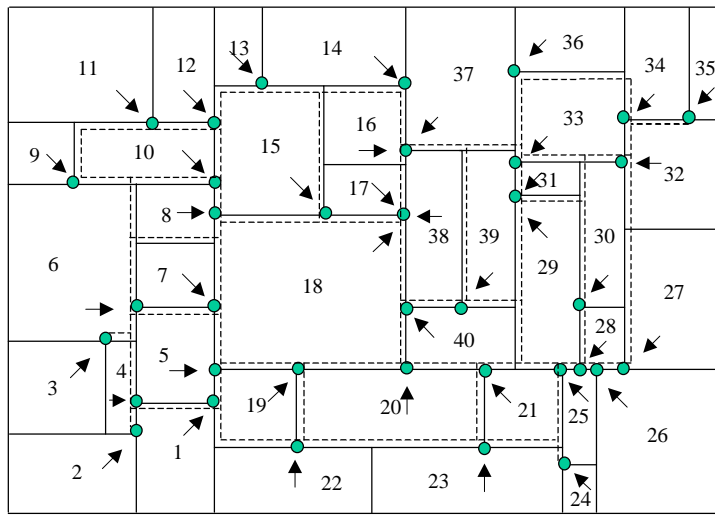
Tam 30 – 4



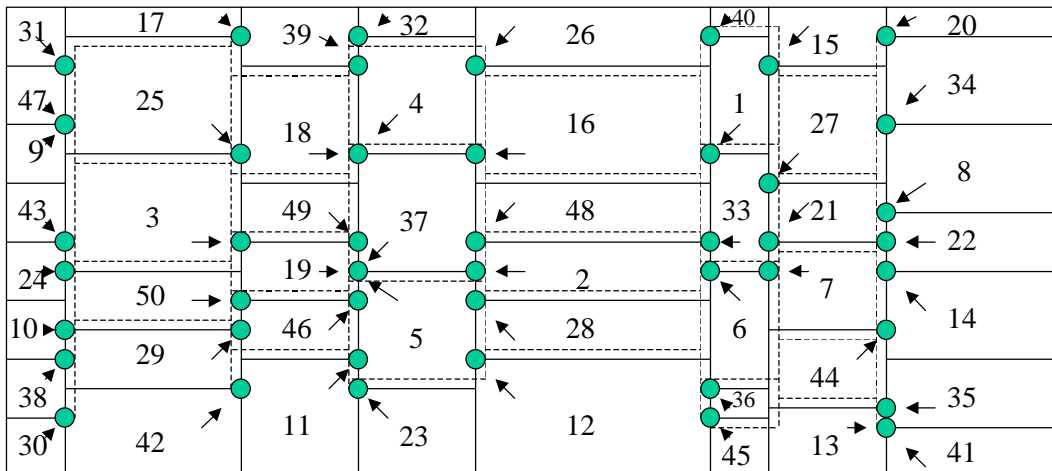
Arapoglu et al. 40 -1



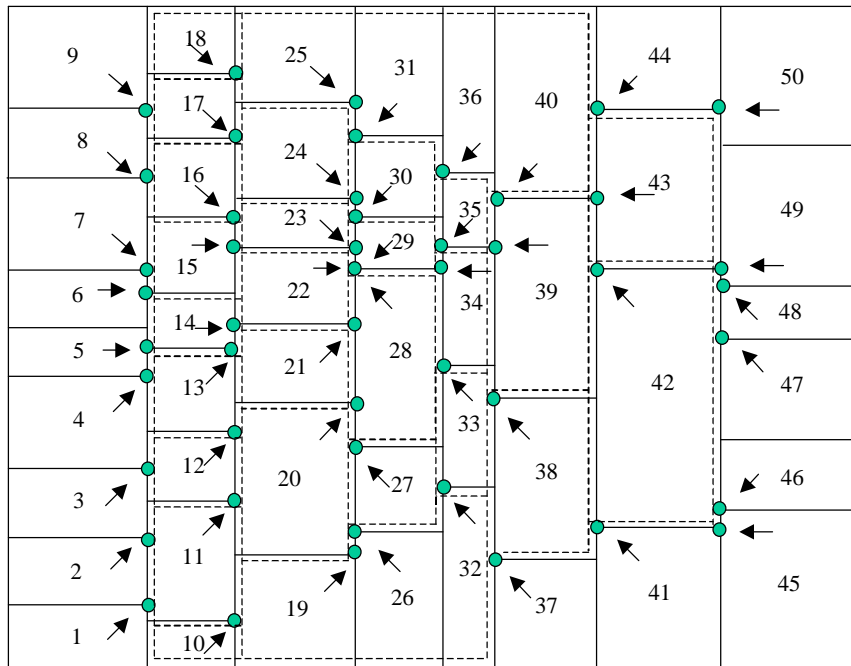
Arapoglu et al. 40 -2



Arapoglu et al. 50 -1



Arapoglu et al. 50 -2



Arapoglu et al. 60

