A HYBRID ALGORITHM FOR OPTIMISING FACILITY LAYOUT

I.A. Tasadduq[1*], M.H. Imam[2] & A. Ahmad[3]

[1]Department of Computer Engineering
Umm Al-Qura University
Makkah, Saudi Arabia
iatasadduq@uqu.edu.sa

[2]Department of Civil Engineering
Umm Al-Qura University
Makkah, Saudi Arabia
mmimam@uqu.edu.sa

[3]Systems Design Engineering
University of Waterloo
Waterloo, Canada
arahim@uwaterloo.ca

ABSTRACT

Despite the reported effectiveness of analytical algorithms in facility layout planning, a detailed literature survey suggests a lack of new analytical methods in recent years. This paper focuses on open space facilities layout planning that involves modules with constant aspect ratios. We propose a construction-cum-improvement algorithm involving a novel combination of a boundary search-based heuristic placement and steepest descent-based analytical improvement. In the construction phase, the algorithm places a new module at the optimal location on the boundary of a previously constructed cluster of modules. In the improvement phase, the algorithm alternates between boundary search and steepest descent moves until it converges to a local optimum. Experiments with well-known test problems indicate that the proposed algorithm produced solutions superior both to published results and to those produced by VIP-PLANOPT, a popular, oft-cited and commercially available layout planning and optimisation software.

OPSOMMING

Ten spyte van die berigte effektiwiteit van analitiese algoritmes met betrekking tot fasiliteitsuitlegbeplanning, dui ‘n gedetailleerde literatuurstudie op ‘n tekort aan nuwe analitiese metodes. In hierdie studie word daar op oop-area fasiliteitsuitlegbeplanning, wat modules met konstante aspek verhouding behels, gefokus. ‘n Konstrukise-en-verbeterings algoritme, wat ‘n nuwe kombinasie van grenssoek gebaseerde heuristiese plasing en steilste gradiënt gebaseerde analitiese verbetering insluit, voorgestel. In die konstruksiefase plaas die algoritme ‘n nuwe module by die optimale posisie op die grens van ’n voorheen geboude groep van modules. Die verbeteringsfase wissel tussen grenssoek- en steilste gradiënt bewegings totdat die algoritme konvergeer tot ‘n plaaslike optimum. Eksperimente met welbekende toetsprobleme dui daarop dat die voorgestelde algoritme beter resultate as gepubliseerde data en dié van VIP-PLANOPT (‘n populêre, dikwels na verwysde, kommersiële fasiliteitsbeplanning sagteware) behaal.

* Corresponding author

1 INTRODUCTION

Facilities layout planning (FLP) involves the allocation of space to activities [1,2]. It is a difficult combinatorial optimisation problem that has received considerable attention from researchers. Some of the more recent work includes [3-10]. Although FLP has been studied extensively with reference to industrial facility layout design, it has applications in various fields of engineering such as machine layout [11], transportation and town planning [12], very large-scale integration (VLSI) design [13], macrocell placement [14], and architectural floor plan design [10].

The most common genre of FLP involves a finite number of rectangular building blocks or modules M_i $(i = 1,2 \ldots N)$, representing various activities or functional units such as departments, machines, rooms, cells, activities, or spaces. The objective is to minimise the cost of inter-module flow by placing all the modules on the packing space without overlaps, in such a manner that the edges of M_i are parallel to the x and y axes respectively. It is a well-known NP-complete problem; and so a verifiably optimal solution cannot be known even for modest size problems [15-17]. Conceivably, published research in this area relies largely on comparing the performance of new algorithms with solutions reported for previously published algorithms, without any attempts to obtain verifiably global optima.

The objective of this paper is to present an effective algorithm for solving a special class of FLP that involves hard modules of unequal dimensions in open space: i.e., with no user-specified boundary constraints. This is in contrast with existing algorithms for solving FLP that involve placing soft modules within a user-specified boundary, which permits modification in the module dimensions within user-specified limits on the aspect ratios. Indeed, such a type of FLP that involves soft modules with a user-specified enclosing boundary has its own role and applications [18]. However, there are numerous applications where alterations in the module dimensions are not permitted [18]. For instance, in a machine layout design application, the dimensions of a machine cannot be changed to fit into a given space. Similarly, many plants have various units that have fixed dimensions that cannot be modified; the same is true with macrocell placement and architectural applications. Further, the requirement of an enclosed space is not always critical, as often there is enough empty space in the land proposed for a facility and minimising the operational costs is more important than simple space use. In short, the FLP with hard modules in open space is an important class of problem from the standpoints of both research and application. Although this important class of FLP has been addressed quite well in the literature, there has been a relative dearth of research over the previous decade or so.

The proposed layout optimisation algorithm is a novel combination of heuristic cluster boundary search in the construction phase and analytical steepest descent search in the improvement phase. This adaptive heuristic boundary search algorithm also employs a near-optimality hypothesis and a definition of local optima for reducing the otherwise infinite search space to tractable limits. It is to be noted that the analytical steepest descent for layout improvement has been employed in the past [19,20]. In addition, the heuristic cluster boundary search using a corner search has also been reported [1,2,21]. However, little research can be found on synergistically employing the two techniques in the same algorithm – an option that offers significant promise. Simulation results presented in this paper demonstrate that the proposed technique is more effective than all previously published algorithms.

The rest of the paper is organised as follows: Section 2 provides a discussion on various classes of FLP, as well as past research in this area. Section 3 defines the problem, along with a mathematical formulation of the problem. Section 4 provides a detailed description of the proposed algorithm. Section 5 provides results and discussions. Section 6 concludes the paper with some interesting future research directions.

2 LITERATURE SURVEY

There are various classes of FLP based on size, shape, and flexibility of the modules, as well as the layout space and dynamicity. For instance, the modules in FLP may be of identical shape and size. Such problems are usually referred to as Equal Area FLP (EA-FLP), where a certain area is divided into identical cells and each module is assigned to one of the cells, making it a cell assignment problem. Several solution approaches have been used to tackle EA-FLP. One popular approach is to treat it as a quadratic assignment problem (QAP) [22]. James, Rego and Glover [23] present an example of solving EA-FLP using Tabu search. Other methods for solving this type of problem are also reported in the literature [24-27].

In contrast, modules in an FLP may be of unequal area. In the case of unequal area FLP (UA-FLP), modules may be either 'soft' or 'hard'. A soft module has a pre-specified area with variable aspect ratio, where the aspect ratio of a soft module is defined as the ratio of the length along Y-axis to its length along X-axis. We refer to this class of problems as soft module unequal area FLP (SUA-FLP). There are several published articles on UA-FLP that involve only soft modules. For instance, the idea of using an ant system for optimally placing soft modules in a given rectangular boundary has been reported in [28]. In addition, a heuristic applicable to both static and dynamic FLP has been reported in [29]. A hybrid algorithm involving slicing tree and Tabu search is reported in [30,31].

Problems that involve only 'hard' modules will be referred to as 'hard module unequal area FLP' (HUA-FLP). In these problems, module areas and aspect ratios are pre-specified. Conceivably, these harder-to-solve problems have received relatively little attention from the research community. Nevertheless, this class of problems is of particular interest from the perspectives of both theoretical research and practical applications.

Another way of classifying FLP is based on the static and dynamic nature of the layouts. In static FLP (SFLP), the material flows between modules are assumed to be constant for the planning horizon. In contrast, in dynamic FLP (DFLP), flows between departments vary during the planning horizon. Several approaches to solving these problems have been reported, including [6,29,32-34]. However, the basic optimisation approach remains the same for both SFLP and DFLP. For instance, McKendall and Hakobyan [29] report an efficient technique for both SLFP and DFLP based on the pioneering cluster boundary search technique proposed by Imam and Mir [35].

Various FLP problems impose a boundary constraint such that the resultant layout is enclosed in a pre-specified area. The boundary constraints in some applications may be specified as a single row of modules of constant width [36,37], as multiple rows [38], or as a rectangle [39]. In contrast, numerous FLP applications do not impose any boundary constraint and permit a layout design in an open space [1,10-14,40].

The proposed algorithm is directed at attacking the open space unequal area hard module facility layout problems (OUH-FLP). It is interesting to note that OUH-FLP has some similarities with the popular cutting-stock problem (CSP) and the two-dimensional bin-packing problem (2D-BPP), in the sense that all these problems are also geared towards optimising some sort of space use [41]. For instance, the cutting-stock problem is an NP-complete optimisation problem, which is essentially reducible to the knapsack problem [42]. In this problem, a given shape or design is required to be cut out of parent material such that trim loss or the amount of parent material wasted is minimised [43]. The FLP problem is quite similar to the cutting-stock problem from the aspect of minimising the unused space/material. As such, there are some similarities between problem-solving approaches for FLP, CSP, and 2D-BPP. Despite these similarities, it should be noted that FLP also involves costs associated with moving materials from one facility to another, which differentiates FLP from the general CSP and 2D-BPP.

As described earlier, our proposed algorithm is focused on solving the ubiquitous OUH-FLP. In the past, OUH-FLP has been solved using various techniques ranging from analytical

methods to evolutionary algorithms. Such techniques include the cluster boundary search algorithm reported in [35], analytic annealing [14], optimisation through controlled convergence [44], and nonlinear programming [19]. Notably, a novel hybrid optimisation approach by synergistically employing simulated annealing and analytical optimisation was reported in [40]. Recently, the analytical cluster boundary search method by [35] has been adapted through a perpetual loop for searching better solutions by using a new module sequence in each iteration [29].

Despite the reported efficiency and efficacy of such analytical algorithms, our extensive literature survey indicates that no new analytical methods for solving OUH-FLP have been published since Mir and Imam [40], with the exception of McKendall and Hakobyan [29]. The thrust of the limited published research in OUH-FLP is towards employing evolutionary algorithms or heuristics. Among such recent algorithms, simulated annealing has been successfully applied to solve OUH-FLP [45]. Nevertheless, the recent evolutionary approaches used to solve OUH-FLP are based largely on genetic algorithms (GA). Kado [46] provides a good survey of some of the earlier work that addresses OUH-FLP using GA and some benchmark problems reported in the literature. The OUH-FLP problem was also tackled using a modified GA that the authors termed a 'co-evolutionary algorithm' [47].

Among the evolutionary algorithms, memetic algorithm (MA) is very promising for solving a wide variety of optimisation problems [48]. MA is a form of hybrid global-local heuristic search methodology, where global search represents a GA while the local search resembles a meme [48, 49]. MAs are known to provide greatly improved exploratory power and computational efficiency [50]. A good account of such MAs is presented in [51]. Recently, MA has also been used to solve the general QAP [52]. However, the use of MAs in solving OUH-FLP has not received much attention in the literature – with some recent exceptions, such as an effective MA reported in [8]. Nevertheless, this important application area is still open to more effective and efficient analytical and heuristic algorithms. This paper is an effort in this direction, and proposes a hybrid analytical-cum-heuristic algorithm.

3 PROBLEM STATEMENT

Let there be N modules of arbitrary but fixed dimensions to be placed at their optimal positions in the Euclidean plane without any overlaps, such that a given cost function ζ is minimised. The position of a module M_i is defined by the coordinates of its centroid (x_i, y_i). Let (L_i, W_i) denote the length and width of module i along the X- and Y-axes respectively. Let f_{ij} represent the cost of inter-module flow per unit distance between modules M_i and M_j and δ_{ij} be the inter-module distance measured between the centroid of modules M_i and M_j. The cost function ζ is defined as follows:

$$minimise\ \zeta(x_1, y_1, x_2, y_2, \dots, x_N, y_N) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} f_{ij}\delta_{ij} \tag{1}$$

The inter-module distance could be any of the distance norm, including Manhattan (rectilinear), Euclidean, or squared Euclidean. In order to avoid the overlapping of modules, any overlap area A_{ij} between two modules M_i and M_j should be restricted to zero. This overlap A_{ij} is defined in the following using [40]:

$$A_{ij} = \lambda_{ij}(\Delta X_{ij})(\Delta Y_{ij}) \tag{2}$$

where

$$\Delta X_{ij} = \left(\frac{L_i + L_j}{2}\right) - |x_i - x_j| \tag{3}$$

$$\Delta Y_{ij} = \left(\frac{W_i + W_j}{2}\right) - |y_i - y_j| \tag{4}$$

$$\lambda_{ij} = \begin{cases} -1 & for\ \Delta X_{ij} \le 0\ and\ \Delta Y_{ij} \le 0 \\ +1 & otherwise \end{cases} \tag{5}$$

A positive value of A_{ij} indicates that there is an overlap between modules M_i and M_j. In the

light of this discussion and Equations (1) to (5), the OUH-FLP problem can now be formulated as:

$$\begin{aligned} & minimise\ \zeta(x_1, y_1, x_2, y_2, \dots, x_N, y_N) \\ subject\ to\quad & \\ & A_{ij} \leq 0; \quad i = 1,2, \dots, N-1; j = i+1\ to\ N \end{aligned} \tag{6}$$

4 PROPOSED ALGORITHM

We propose a hybrid algorithm involving dual hybridisations: the construction-cum-improvement hybrid and the analytical-cum-heuristic hybrid. We refer to the proposed algorithm as the adaptive cluster boundary search (ACS) algorithm. The algorithm is adaptive in two ways: the first phase is a constructive algorithm in which the search space adapts to the contour of the boundary formed by the layout obtained in the previous iteration. The second phase is an improvement type algorithm that adapts to the values of the derivatives of the placed modules. These derivatives represent error in the placement from the optimal placement, and the modules are moved to minimise this error. This will be explained further in Section 4.4.

In the following section, we provide a detailed description of the main components of the proposed algorithm. Pseudo-code and a flow chart of the proposed ACS algorithm are given in Appendix A and Figure 1 respectively.

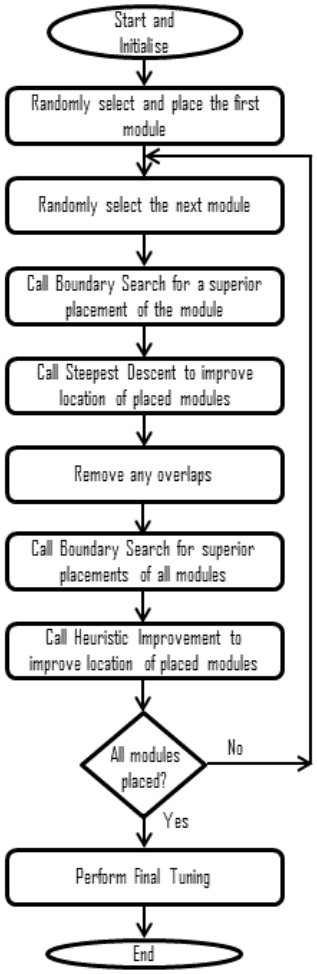


Figure 1: Flow chart of the proposed algorithm

4.1 Initialisation

The constructive part of the algorithm starts with an initial layout consisting of two modules. This layout is obtained by randomly selecting a module as the first incoming module (IM_1) and placing it at an arbitrary position. Once placed, this module will be called 'placed module' (*PM*) and will be denoted by (PM_i, x_i, y_i), where *i* identifies the placed module at position (x_i, y_i). This PM_i is then added to the set of placed modules denoted by ξ_{PM}. This is followed by randomly selecting another module as the second incoming module (IM_2) and placing it optimally on the boundary of the *PM*. Such a placement that results in the minimum cost position on the boundary of the *PM*s will be referred to as 'initial boundary placement' (*IBP*), inspired by the idea described in [35]. Since at this stage only two *PM*s have been placed, the partial layout is also the global optimum layout; there is thus no need to go beyond the *IBP* to perform an improvement cycle until there are more than two *PM*s.

4.2 Main algorithm

The main ACS algorithm loop starts with the selection of the third module. Each time in this loop, an incoming module *IM* is randomly selected from the list of candidate modules (CM). This is followed by *IBP* to determine the best possible position on the boundary of the layout consisting of a contiguous cluster of *PM*s. The algorithm ensures a contiguous cluster of *PM*s with no splits by merging any mini-clusters formed during placements.

After the placement of the *IM* through *IBP*, the improvement cycle starts. The improvement cycle consists of several steps: the steepest descent-based analytical improvement described in Section 4.4, the removal of any overlaps, the heuristic boundary search for placement of any modules moved during overlap removal, and the heuristic improvement described in Section 4.5.

The main loop continues until all modules have been placed.

4.3 Heuristic boundary search

The heuristic boundary search explores placement of an *IM* or PM_j, as the case may be, only at the vacant four corners of the placed module PM_x. This heuristic reduces the otherwise infinite solution space to tractable limits, as for a given IM there are only *O(n)* possible locations. More specifically, for given *i* modules in the cluster, there are a maximum of 12*i* possible locations to explore, which is a very loose upper bound.

In the proposed algorithm, the heuristic boundary search is called twice: first, to place every incoming module (*IM*); and second, to find a better location for a placed module (PM_π) after the application of the steepest descent improvement technique.

4.4 Analytical improvement

Once the best position for an *IM* has been found on the cluster of placed modules using heuristic boundary search, the analytical improvement through steepest descent is carried out on all *PM*s. For this purpose, the slopes of steepest descent for all *PM*s are determined as follows:

$$S_{k,x} = \frac{\partial \zeta}{\partial x_k},\ S_{k,y} = \frac{\partial \zeta}{\partial y_k} \tag{7}$$

$$|S_k| = \sqrt{S_{k,x}^2 + S_{k,y}^2} \tag{8}$$

where $S_{k,x}$ and $S_{k,y}$ are the x and y components of slope of the k^{th} *PM* and $|S_k|$ represents the magnitude of this slope. Following the decreasing order of$|S_k|$, each PM is moved in the direction of the steepest descent, ignoring any overlaps, until the cost does not decrease any further. Any overlapping modules, except for the module moved by the analytical improvement procedure, are placed at superior positions using heuristic boundary search. If the new layout does not result in an improved cost, the improvement procedure is backtracked, the module with the next highest slope is moved, and the procedure is repeated. If the move is successful, the slopes are calculated again and the analytical

improvement procedure is repeated. Once all modules have been considered, or no further improvement is possible, the heuristic boundary search is applied to all modules. This cycle of analytical improvement through steepest descent and heuristic boundary search is repeated until no gains are possible.

4.4.1 Adaptivity

The presented algorithm is highly adaptive. First, the algorithm has a constructive phase and an improvement phase, which are both adaptive. The adaptive nature is controlled by the parameters of OUH-FLP, which are addressed in this paper. There are only six distinct parameters: (a) cost matrix, (b) module dimensions, (c) positions of modules, (d) boundary of placed modules, (e) cost of the layout, and (f) the derivatives of the cost with respect to the x and y coordinates of the layout.

In the constructive phase, the search space for each incoming module adapts to the contours of the boundary obtained in the previous iteration. Here the parameter that controls the algorithm is the boundary of the layout obtained in the previous iteration. The boundary data is kept in a digital form. The module is guided by the boundary data to move and search for its own optimal location along the layout boundary. Thus the search space in the constructive part of the algorithm is adaptive.

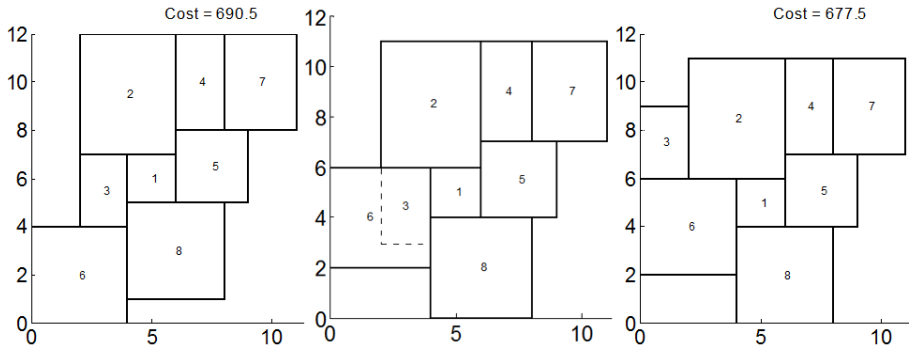
In the improvement part of the algorithm, a module is selected for movement, and is moved in the direction of the steepest descent that is based on the derivatives of the cost function with respect to the x and y coordinates. Both the selection of the module and its movement are adaptive. The derivatives represent an estimate of error in the placement of the module in a given layout. Optimality is achieved when all derivatives are zero and a layout becomes optimal (at least locally), and no module can be moved to obtain a better layout. Considering this, we use the derivatives and select the module with the highest derivative. The module is then moved in the direction of steepest descent, which again is a function of derivatives. Thus the parameter making the algorithm adaptive is the derivative representing the error in the placement of a module. It is similar to several applications where the estimate of error in the previous iteration is used in deciding the move. Such algorithms are referred to as adaptive algorithms, such as the well-known ‘adaptive coordinate descent method’ [53] and the ‘adaptive algorithms for deterministic and stochastic differential equations’ [54].

4.4.2 An example of analytical improvement

This analytical improvement procedure and the subsequent overlap removal are explained here through the example given in Figure 2. This figure shows a cluster of eight *PM*s formed after applying a heuristic boundary search on IM_8. Since the best position of IM_8 on the boundary of the cluster has been obtained, analytical improvement through steepest descent procedure will be applied to each of the *PM*s. In the example shown in Figure 2, M_6 has the highest value of S_k. This module is therefore moved first in its direction of steepest descent, which is vertically upwards. The optimal position is shown in Figure 2(b), while ignoring the overlap with M_3. To remove any overlap, heuristic boundary search is applied to M_3 for determining its best position on the boundary of the cluster, resulting in the layout shown in Figure 2(c). It is to be noted that the analytical improvement may result in an overlap of multiple modules. In such a case, all overlapping modules, except for the module moved by the analytical improvement procedure, are placed at superior positions using heuristic boundary search.

4.5 Heuristic improvement

Our heuristic improvement is inspired by the ideas explored in [55]. It involves minor horizontal and vertical perturbations in the position of each module present in the cluster, by considering one module at a time. These perturbations are retained only when these result in cost improvement. Furthermore, to reduce computational complexity, we perform these heuristic improvements in a slightly greedy manner. For instance, a useful rightward perturbation would mean ignoring any leftward perturbation or vice versa; although vertical perturbations are still explored.



(a) Before analytic improvement (b) Steepest descent move of Module 6 (c) Module 3 moved to optimal boundary location

Figure 2: Example of the application of the analytic improvement procedure

4.6 Final tuning

This final tuning was performed once only at the end of all the iterations of analytical and heuristic improvements. It involves detecting any super modules, where adjacent modules perfectly align along one side, either vertically or horizontally. The heuristic improvement described in Section 4.5 is then applied to such super modules. Our simulation studies indicate that the overall improvement due to this final tuning was only about 0.1–0.2 per cent of the cost of the final solution. However, since the computational cost of this simple fine tuning is almost negligible – in the range of a few milliseconds in total – it is a worthwhile investment.

5 RESULTS AND DISCUSSIONS

The proposed algorithm was implemented using Matlab, and was used to solve the benchmark problems listed in Table 1, using a 2.0 GHz quad-core computer running on Windows 7 with a memory of 8 GB. The data for the benchmark problems can be found in [56]. Comparison of layouts obtained from the proposed ACS algorithm with the verifiably best available published results and those obtained from VIP-PLANOPT 10 [57], a popular, oft-cited commercially available layout planning and optimisation software, are presented here. Recently, results from VIP-PLANOPT have often been employed as benchmarks for testing the effectiveness of new algorithms in facility layout planning [1,5,8,29,58-60]. Thus we deem it appropriate to include it in our comparative analysis.

It should be noted here that several solutions published in the literature are without the necessary data or layout diagram of the claimed superior layout, barring any possibility of independently authenticating those solutions. We made concerted efforts to obtain verifiable data for those published solutions by personally communicating with authors who claimed those good solutions. When no necessary data was available, such published claims were excluded from our comparative analyses. Table 1 provides a summary of these comparative analyses, clearly indicating the superiority of the proposed algorithm.

5.1 Summary of results

A summary of results is provided in Table 1. In this table, each row provides a separate benchmark problem; the columns give the problem ID, the problem size in terms of the number of modules, the cost function, the best verifiable published result, the best solution obtained using VIP-PLANOPT 10, and the best solution obtained through the proposed ACS algorithm. The distance norm used in all the problems is rectilinear, except for problem numbers *5*, *6* and *12*, where the squared Euclidean norm has been used, and problem 13, where the Euclidean norm has been used.

It can be seen from Table 1 and from Figures 3 to 8 that the solutions obtained by the proposed technique are better than all previously published analytical, heuristic, evolutionary, and memetic algorithms, as well as commercially available layout optimisation software. Data relevant to the layouts optimised using the proposed ACS algorithm, such as coordinates of the lower left corners and problem dimensions, etc., are available from [56].

Table 1: Summary of results

	Problem ID	Size (Modules)	Cost Function	Best Published	Best from PLANOPT	ACS	
1	OUH-003-Ixx [57]	3	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	270.00 [1]	270.00	270.00	**
2	OUH-004-Dxx [61]	4	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	1,510.00 [1]	1,510.00	1,510.00	**
3	OUH-006-Dxx [61]	6	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	3,314.80 [1]	3,379.00	3,274.00	***
4	OUH-008-Dxx [61]	8	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	-	10,468.00	10,468.00	**
5	OUH-008-IMx [62]	8	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	689.50 [1]	692.50	676.50	***
6	OUH*f*-008-EOS [57]	8	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	757.50 [1]	763.50	762.23	*
7	OUH-010-Dxx [61]	10	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	19,279.00 [1]	19,162.00	18,488.59	***
8	OUH-011-IMx [62]	11	$2\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij} + Area$	2,714.19 [2]	2,730.70	2,626.48	***
9	OUH-012-Dxx [61]	12	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	-	43,180.50	41,257.19	***
10	OUH-020-Mix [63]	20	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	1,151.40 [3]	1,157.00	1,140.00	***
11	OUH-020-CHM [64]	20	$\sum_{i=1}^{N}\sum_{j=1}^{N} f_{ij}\delta_{ij} + Area$	140.00 [1]	128.00	119.00	***
12	OUH-028-Mix [63]	28	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	8,640.00 [4]	6,447.25	6,289.64	***
13	OUH-050-EOS [57]	50	$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} f_{ij}\delta_{ij}$	71,291.40 [3]	78,224.70	71,151.41	***

***ACS surpassed the best; **ACS achieved the best; *ACS comparable to the best

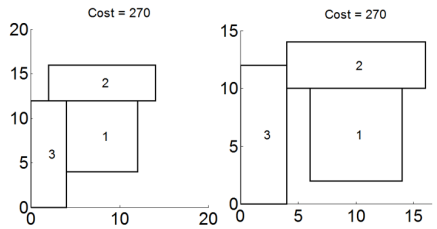


Figure 3: Two possible optimal layouts of OUH-003-Ixx

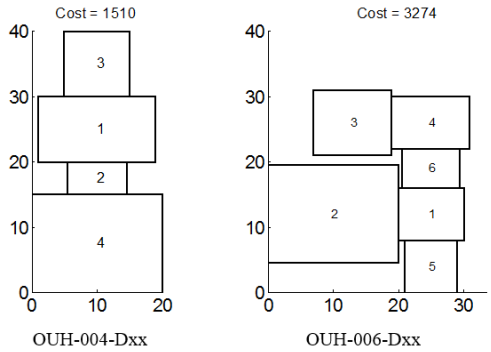


Figure 4: Optimal layouts for 4 and 6 module benchmark problems

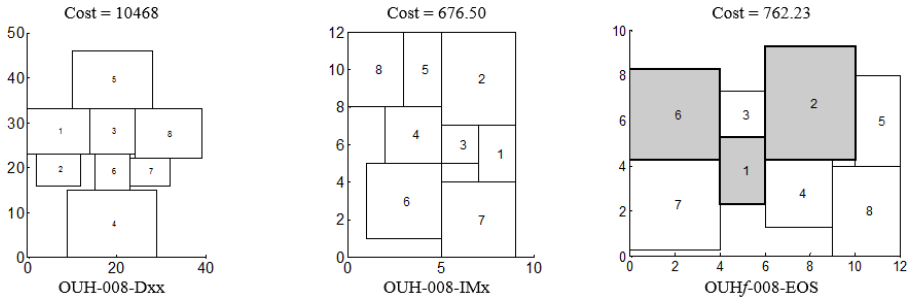


Figure 5: Optimal layouts for 8 module benchmark problems

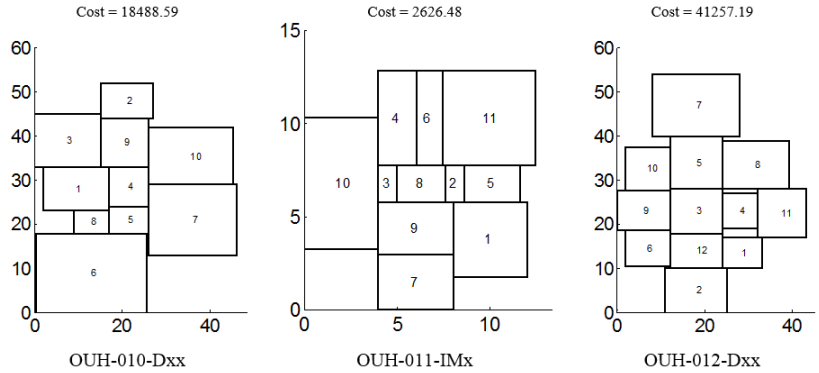


Figure 6: Optimal layouts for 10, 11, and 12 module benchmark problems

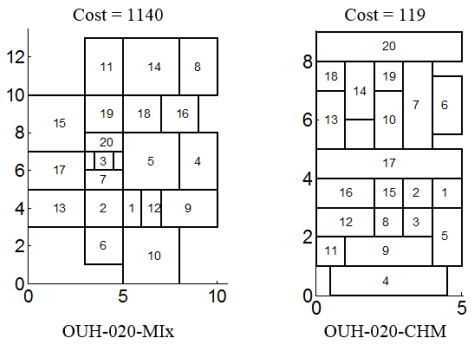


Figure 7: Optimal layouts for 20 module benchmark problems

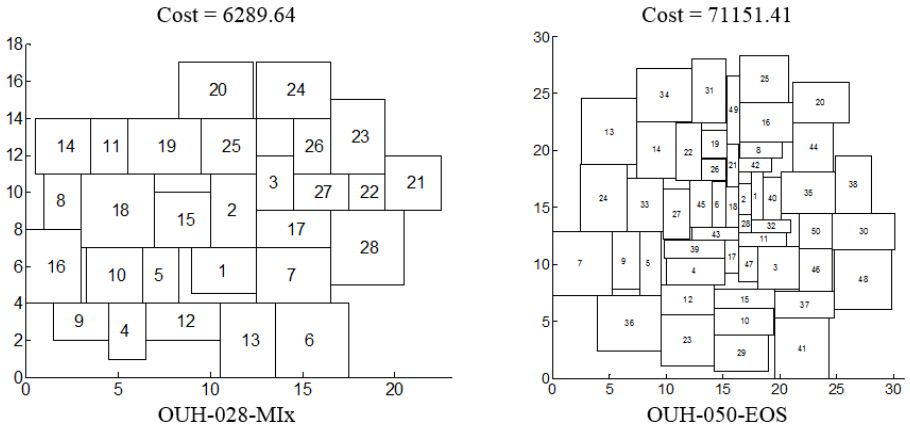


Figure 8: Optimal layouts for 28 and 50 module benchmark problems

6 CONCLUSION

This paper presents an effective algorithm for solving a special class of FLP that involves unequal area hard modules in open space. The proposed construction-cum-improvement algorithm is a novel combination of analytical steepest descent and heuristic cluster boundary search. We also provide an extensive literature review related to open-space hard module FLP, along with a compilation of published problems. Test results demonstrate that the proposed algorithm resulted in significant improvements over published techniques and commercially available software. This research is expected to renew the interest of the research community in this important research domain. In future, we want to explore the idea of a more exhaustive final tuning by considering not only the super modules, but also their subsets, which may result in more promising solutions. Some fuzzy logic rules for ruling out less promising placements in the heuristic cluster boundary search may also help to improve the overall efficiency.

REFERENCES

[1] **Ahmad, A.R., Basir, O., Hassanein, K. and Imam, M.H.** 2006. An effective module placement strategy for genetic algorithms based layout design. *International Journal of Production Research,* 44, pp. 1545-1567.

[2] **Ahmad, A.R.** 2005. *An intelligent expert system for decision analysis & support in multi-attribute layout optimization.* PhD thesis, University of Waterloo, Canada. Available online: https://uwspace.uwaterloo.ca/handle/10012/785

[3] **Kulturel-Konak, S. and Konak, A.** 2013. Linear programming based genetic algorithm for the unequal area facility layout problem. *International Journal of Production Research,* 51, pp. 4302-4324.

[4] **Hosseini-Nasab, H. and Emami, L.** 2013. A hybrid particle swarm optimisation for dynamic facility layout problem. *International Journal of Production Research,* 51, pp. 4325-4335.

[5] **Xiao, Y., Seo, Y. and Seo, M.** 2013. A two-step heuristic algorithm for layout design of unequal-sized facilities with input/output points. *International Journal of Production Research,* 51, pp. 4200-4222.

[6] **McKendall, A.R. and Liu, W.-H.** 2012. New Tabu search heuristics for the dynamic facility layout problem. *International Journal of Production Research,* 50, pp. 867-878.

[7] **Yang, C.-L., Chuang, S.-P. and Hsu, T.-S.** 2011. A genetic algorithm for dynamic facility planning in job shop manufacturing. *The International Journal of Advanced Manufacturing Technology,* 52, pp. 303-309.

[8] **Tasadduq, I.A., Imam, M.H. and Ahmad, A.-R.** 2011. A novel metasearch algorithm for facility layout problems. *In: 41st International Conference on Computers & Industrial Engineering.*

[9] **Anjos, M. F., & Liers, F**. 2012. Global approaches for facility layout and VLSI floorplanning. In *Handbook on Semidefinite, Conic and Polynomial Optimization* (pp. 849-877). Springer US.

[10] **Chung, J. and Tanchoco, J.M.H.** 2010. Layout design with hexagonal floor plans and material flow patterns. *International Journal of Production Research,* 48, pp. 3407-3428.

[11] **Moslemipour, G., Lee, T.S. and Rilling, D.** 2012. A review of intelligent approaches for designing dynamic and robust layouts in flexible manufacturing systems. *The International Journal of Advanced Manufacturing Technology,* 60, pp. 11-27.
[12] **Saif, M.A. and Imam, M.H.** 2004. Computer aided layout optimisation model for minimizing traffic volume. *In*: Topping, B.H.V. and Soares, C.A.M. (eds) *Proceedings of the Fourth International Conference on Engineering Computational Technology*. Paper 49. Stirlingshire, UK: Civil-Comp Press.
[13] **LaPaugh, A.S.** 2010. VLSI Layout Algorithms. *In:* Atallah, M.J. and Blanton, M. (eds) *Algorithms and theory of computation handbook: Special topics and techniques.*Chapman and Hall/CRC.
[14] **Mir, M. and Imam, M.H.** 1996. Analytic annealing for macrocell placement problem. *Computers & Electrical Engineering,* 22, pp. 169-177.
[15] **Ahmad, A.R., Basir, O., Hassanein, K. and Imam, M.H.** 2005. A hierarchical placement strategy for generating superior layout decision alternatives. *International Journal of Operations and Quantitative Management,* 11, pp. 261-280.
[16] **Garey, M.R. and Johnson, D.S.** 1979. *Computers and intractability*. NY: W.H. Freeman Press.
[17] **Sahni, S. and Gonzalez, T.** 1976. P-Complete approximation problems. *Journal of the ACM,* 23, pp. 555-565.
[18] **Tompkins, J.A., White, J.A., Bozer, Y.A. and Tanchoco, J.M.A.** 2010. *Facilities planning*. New York: John Wiley.
[19] **Mir, M. and Imam, M.H.** 1989. Optimal placement for hierarchical VLSI layout design. *Microprocessing and Microprogramming,* 25, pp. 177-182.
[20] **Mir, M. and Imam, M.H.** 1990. A gradient based method for module placement. *Computers and Electrical Engineering,* 16, pp. 109-113.
[21] **Welgama, P.S. and Gibson, P.R.** 1993. A construction algorithm for a machine layout problem with fixed pick-up and drop-off points. *International Journal of Production Research,* 31, pp. 2575-2590.
[22] **Ramkumar, A.S. and Ponnambalam, S.G.J.N.** 2009. A new iterated fast local search heuristic for solving QAP formulation in facility layout design. *Robotics and Computer-Integrated Manufacturing,* 25, pp. 620-629.
[23] **James, T. ,Rego, C. and Glover, F.** 2009. A cooperative parallel Tabu search algorithm for the quadratic assignment problem. *European Journal of Operational Research,* 195, pp. 810-826,.
[24] **Misevicius, A.** 2003. A modified simulated annealing algorithm for quadratic assignment problem. *Informatica,* 14, pp. 497-514.
[25] **Talbi, E.G., Roux, O., Fonlupt, C. and Robillard, D.** 2001. Parallel ant colonies for quadratic assignment problem. Future Generation. *Comput Syst,* 17, pp. 441-449.
[26] **Ahuja, R.K., Orlin, J.B. and Tiwari, A.** 2000. A greedy genetic algorithm for the quadratic assignment problem. *Computers and Operations Research,* 27, pp. 917-934.
[27] **Raoot, A.D. and Rakshit, A.** 1994. A fuzzy heuristic for the quadratic assignment formulation to the facility layout problem. *International Journal of Production Research,* 32, pp. 563-581.
[28] **Wong, K.Y.** 2010. Applying ant system for solving unequal area facility layout problems. *European Journal of Operational Research,* 202, pp. 730-746.
[29] **McKendall, A.R. and Hakobyan, A.** 2010. Heuristics for the dynamic facility layout problem with unequal-area departments. *European Journal of Operational Research,* 201, pp. 171-182.
[30] **Scholz, D., Jaehn, F. and Junker, A.** 2010. Extensions to STaTS for practical applications of the facility layout problem. *European Journal of Operational Research,* 204, pp. 463-472.
[31] **Scholz, D., Petrick, A. and Domschke, W.** 2009. STaTS: A slicing tree and Tabu search based heuristic for the unequal area facility layout problem. *European Journal of Operational Research,* 197, pp. 166-178.
[32] **Sahin, R., Ertogral, K. and Türkbey, O.** 2010. A simulated annealing heuristic for the dynamic layout problem with budget constraint. *Computers and Industrial Engineering,* 59, pp. 308-313.
[33] **Ulutas, B.H. and Islier, A.A.** 2009. A clonal selection algorithm for dynamic facility layout problems. *Journal of Manufacturing Systems,* 28, pp. 123-131.
[34] **Dong, M., Wu, C. and Hou, F.** 2009. Shortest path based simulated annealing algorithm for dynamic facility layout problem under dynamic business environment. *Expert Systems with Applications,* 36, pp. 11221-11232.
[35] **Imam, M.H. and Mir, M.** 1998. Cluster boundary search algorithm for building-block layout optimization. *Advances in Engineering Software,* 29, pp. 165-173.
[36] **Samarghandi, H., Taabayan, P. and Jahantigh, F.F.** 2010. A particle swarm optimization for the single row facility layout problem. *Computers and Industrial Engineering,* 58, pp. 529-534.
[37] **Anjos, M.F. and Yen, G.** 2008. Provably near-optimal solutions for very large single-row facility layout problems. *Optimization Methods and Software,* 24, pp. 805-817.
[38] **Ficko, M., Balic, J., Brezocnik, M. and Pahole, I.** 2010. Solving of floor layout problem in flexible manufacturing system by genetic algorithms. *International Journal of Advanced Intelligence Paradigms,* 2, pp. 354-364.

[39] **Jankovits, I., Luo, C., Anjos, M.F. and Vannelli, A.** 2011. A convex optimisation framework for the unequal-areas facility layout problem. *European Journal of Operational Research,* 214, pp. 199-215.
[40] **Mir, M. and Imam, M.H.** 2001. A hybrid optimization approach for layout design of unequal area facilities. *Computers and Industrial Engineering,* 39, pp. 49-63.
[41] **Ahmad, A.R.** 2014. A new hierarchical placement algorithm for two-dimensional rectangular layout design. *International Journal of Operations and Quantitative Management,* 20, pp. 101-120.
[42] **Garey, M.R. and Johnson, D.S.** 1979. Computers and intractability: An introduction to the theory of NP-completeness. San Francisco: W.H. Freeman.
[43] **Chandrasekaran, J.B.A.R.** 1996. Stock cutting to minimize cutting length. *European Journal of Operational Research,* 88, pp. 69-87.
[44] **Imam, M.H. and Mir, M.** 1993. Automated layout of facilities of unequal areas. *Computers & Industrial Engineering,* 24, pp. 355-366.
[45] **Tam, K.Y.** 1992. A simulated annealing algorithm for allocating space to manufacturing cells. *The International Journal of Production Research,* 30, pp. 63-87.
[46] **Kado, K.** 1995. *An investigation of genetic algorithms for facility layout problems*. Doctoral Thesis, University of Edinburgh.
[47] **Dunker, T., Radons, G. and Westcamper, E.** 2003. A coevolutionary algorithm for a facility layout problem. *International Journal of Production Research,* 41, pp. 3479-3500.
[48] **Moscato, P**. 1989. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, *826*, 1989.
[49] **Ong, Y.-S., Lim, M.H. and Chen, X.** 2010. Memetic computation – past, present & future. *IEEE Computational Intelligence Magazine,* 5, pp. 24-31.
[50] **Goldberg, D.E. and Voessner, S.** 1999. Optimizing global-local search hybrids. *In: Genetic and Evolutionary Computation Conference*, pp. 220-228.
[51] **Molina, D., Lozano, M., Garcia-Martinez, C. and Herrera, F.** 2010. Memetic algorithms for continuous optimisation based on local search chains. *Evolutionary Computation,* 18, pp. 27-63.
[52] **Drezner, Z.** 2008. Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. *Computers and Operations Research,* 35, pp. 717-736.
[53] **Loshchilov, I., Schoenauer, M. and Sebag, M.** 2011. Adaptive coordinate descent. *In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pp. 885-892.
[54] **Moon, K.-S., Szepessy, A., Tempone, R. and Zouraris, G.E.** 2003. Convergence rates for adaptive approximation of ordinary differential equations. *Numerische Mathematik,* 96, pp. 99-129.
[55] **Imam, M.H. and Tasadduq, I.A.** 2010. An extremely simple operation for drastic performance enhancement of genetic algorithms for engineering design optimization. *International Journal of Engineering Science and Technology,* 2, pp. 6630-6645.
[56] **Tasadduq, I.A., Imam, M.H. and Ahmad, A.R.** 2012. *Problem dimensions and best layouts*. Available: http://pami.uwaterloo.ca/pub/rahim/TIA2012.xlsm
[57] **VIP-PLANOPT.** *Engineering Optimization Software*. Available: http://www.planopt.com/
[58] **Oheba, J.** 2012. A new framework considering uncertainty for facility layout problem. Doctoral thesis, University of Manchester, Manchester, UK. Available online: https://www.escholar.manchester.ac.uk/uk-ac-man-scw:181576
[59] **Heragu, S.S.** 2006. *Facilities design*. Lincoln: iUniverse.
[60] **Tompkins, J.A., White, J.A., Bozer, Y.A. and Tanchoco, J.M.A.** 2002. *Facilities planning*. New York: John Wiley.
[61] **Das, S.K.** 1993. A facility layout method for flexible manufacturing systems. *Int. J. Prod. Res.*, 31, pp. 279-297.
[62] **Imam, M.H. and Mir, M.** 1989. Nonlinear programming approach to automated topology optimization. *Computer Aided Design,* 21, pp. 107-115.
[63] **Mir, M. and Imam, M.H.** 1992. Topology optimization of arbitrary-size blocks using a bivariate formulation. *Computer Aided Design,* 24, pp. 556-564.
[64] **Cohoon, J.P., Hegde, S.U., Martin, W.N. and Richards, D.S.** 1991. Distributed genetic algorithms for the floorplan design problem. *IEEE Trans. CAD,* 10, pp. 483-492.

APPENDIX A: Pseudo-Code for the Adaptive Cluster Boundary Search (ACS) algorithm

Let,

ζ	= Cost of the objective function	ξ_{CM}	= Set of candidate modules
TM	= Total number of modules	ξ_{PM}	= Set of placed modules with locations
IM	= Incoming module	ξ_{PMT}	= Temporary Set of *PM*s
PM	= Placed module	ξ_{PMO}	= Set of *PM*s with possible overlap(s)
TPM	= Total number of *PM*s	ξ_{PMS}	= Set of *PM*s with slopes in descending order
(IM, x_{IM}, y_{IM})	= Incoming module placed at (x_{IM}, y_{IM})	χ, π	= A PM_χ at whose corners an IM_π or another PM_π is being placed
TPMO	= Total number of *PM*s with possible overlap(s)	ε	= A small positive number

/Initialization/
Set *TPM* = 0, $\zeta = \infty$
Select $IM \in \xi_{CM}$
Place IM at (0, 0)
Add $(IM, 0,0)$ to ξ_{PM}
Increment *TPM*

Main ( )
For *p* = 2 to *TM*
Select $IM \in \xi_{CM}$
$\pi = p$
Call BoundarySearch ( )
Add (IM, x_{IM}, y_{IM}) to ξ_{PM}
Increment *TPM*
Call ComputeCost ( )
$\zeta = \zeta_{TPM}$
Call SteepestDescent ( )
For *q* = 1 to *TPM*
$\pi = q$
Call BoundarySearch ( )
Call HeuristicImpr ( )

BoundarySearch ( )
For *k* = 1 to *TPM*
$\chi = k$
Call ModulePlacement ( )
If $\zeta_{Min} < \zeta$
$\zeta = \zeta_{Min}$
$\xi_{PM} = \xi_{PMT}$

ModulePlacement ( )
Set $\zeta_{Min} = \infty$, $\xi_{PMT} = \xi_{PM}$
For *i* = 1 to 4
For *j* = 1 to 4
If $i \neq j$: Place *i*th corner of IM_π / PM_π at *j*th corner of PM_χ
Call ComputeCost ( )
If $\zeta_{TPM} < \zeta_{Min}$
$\zeta_{Min} = \zeta_{TPM}$
Save current module placements as ξ_{PMT}
Return ζ_{Min}
Return ξ_{PMT}

SteepestDescent ( )
For *k* = 1 to *TPM*
Compute $S_{k,x} = \frac{\partial \square}{\partial x_k}$, $S_{k,y} = \frac{\partial \square}{\partial y_k}$
Compute $S[k] = \sqrt{S_{k,x}^2 + S_{k,y}^2}$
Sort ξ_{PM} by S in descending order and save in ξ_{PMS}
$\xi_{PMT} = \xi_{PM}$

For *i* = 1 to *TPM*
Do while $\zeta_{TPM} < \zeta$
Move $PM_i \in \xi_{PMS}$ in the direction of *S*[*i*]
Call ComputeCost ( )
Call CheckOverlaps ( )
If *TPMO* ≠ Ø
Call RemoveOverlaps ( )
Call ComputeCost ( )
Save layout in ξ_{PMT}
If $\zeta_{TPM} < \zeta$
$\xi_{PM} = \xi_{PMT}$
$\zeta = \zeta_{TPM}$

HeuristicImpr ( )
For *i* = 1 to π
For *j* = 1 to 4 / *j* = 1: positive *x*; *j* = 2: negative *x*; *j* = 3: positive *y*; *j* = 4: negative y /
Do while $\zeta_{TPM} < \zeta$
Move PM_i in direction "*j*" by ε
Call ComputeCost ( )
Save layout in ξ_{PMT}
If $\zeta_{TPM} < \zeta$
$\xi_{PM} = \xi_{PMT}$
$\zeta = \zeta_{TPM}$

CheckOverlaps ( )
For *i* = 1 to *TPM*
For *j* = *i*+1 to *TPM*
If overlap
Add module to ξ_{PMO}
Increment *TPMO*

RemoveOverlaps ( )
While *TPMO* > 0
Select first module in ξ_{PMO}
$\pi = 1$
Call BoundarySearch ( )
Decrement *TPMO*
Update ξ_{PMO}
ComputeCost ( )
Set ζ_{TPM} = 0
For *i* = 1 to *TPM* - 1
For *j* = (*i* + 1) to *TPM*
Compute $\zeta_{TPM} = \zeta_{TPM} + f_{ij}\delta_{ij}$
Return ζ_{TPM}