ARTICLE

MEGARON 2018;13(4):521-535

DOI: 10.5505/MEGARON.2018.37929

A Modular and Dynamic Evolutionary Algorithm For Architectural Design

Mimari Tasarım İçin Modüler ve Dinamik Bir Evrimsel Algoritma

Nizam Onur SÖNMEZ

ABSTRACT

As we move away from well-defined problem domains, and get closer to more open-ended domains like planning and design, an increase in the complexity of the problems compel the problem-solving behavior to change in a qualitative sense. Consequently, dynamic problem solving strategies appear as one of the requirements for computational design studies. This paper presents a novel multi-objective Evolutionary Algorithm (EA) called the Interleaved EA (IEA) as a problem-solving tool, which incorporates dynamic aspects. It is specific to IEA that one of the objectives leads the evolution until its fitness progression stagnates. As such, IEA enables the use of different settings and operators for each of its objectives, which would be the same for all objectives in a regular EA. This enables the IEA to dynamically adapt its problem setting throughout its progression. We present the specificities of the IEA with an application on a design problem. As the IEA has been developed to assist in design problems, it is examined through the "Architectural Layout Design" problem studied through library buildings, exemplifying an ill-defined, multi-modal, and multi-objective problem. We compare the functioning of the algorithm with regard to, first, a regular rank-based version, for demonstrating the effect of the leading objective approach; secondly, with a popular multi-objective EA (i.e., NSGA2). We discuss how and why IEA can be used and developed further to incorporate domain specific understanding for multi-modal and dynamic design problems.

Keywords: *Automated plan layout development; computational architecture; computational design; evolutionary computation in design; evolutionary design.*

ÖZ

İyi tariflenmiş problem alanlarından planlama ve tasarım gibi kötü-tarifli alanlara doğru geçtiğimizde karşılaştığımız problemlerin karmaşıklığı problem çözme yaklaşımında niteliksel değişiklikler dayatır. Bunun sonucu olarak dinamik problem çözme stratejileri hesaplamalı tasarım çalışmaları için bir gereklilik olarak açığa çıkar. Bu çalışma İçiçe Evrimsel Algoritma (IEA) adında yeni bir çoklu-objektifli Evrimsel Algoritmayı (EA) dinamik yönlere sahip bir problem çözme aracı olarak sunmaktadır. IEA'nın diğer EA'lardan farkı, kullanılan objektiflerden birinin zindelik ilerlemesi duraklayana kadar süreci çeşitli açılardan yönlendirmesi ve daha sonra yönlendirme işlevini bir diğer objektife devretmesidir. Bu şekilde IEA'nın farklı objektifler için farklı ayar ve operatörler kullanması mümkün olmaktadır. Bu sayede IEA problem tanımını işleyişi boyunca dinamik biçimde uyarlayabilmektedir. IEA özel olarak tasarım problemlerinin çizgisel-olmayan, karmaşık karakterine dönük olarak geliştirildiği için bu makalede IEA'nın kendine has özelliklerini kötü-tanımlı, çoklu-modlu ve çoklu objektifli bir problem olan Mimari Plan Düzenlemesi Problemi üzerinden ve kütüphane binaları özelinde sunuyoruz. İlk olarak, IEA'nın işleyiş karakteristiklerini ve başarımını bir sıralama-tabanlı EA versiyonuyla kıyaslayarak yukarıda anlattığımız yönlendirici objektif yaklaşımının sonuçlarını ortaya koyuyoruz. Ardından, işleyiş karakteristiklerini daha derinlemesine yorumlamak üzere IEA'yı popüler bir çoklu-objektifli EA olan NSGA2 ile kıyaslıyoruz. Son olarak spesifik tasarım alanlarına ait bilgiyi işe koşmanın gerekçelerini ve yollarını tartışarak IEA'nın nasıl kullanımlarının olabileceğini ve nasıl daha öte geliştirilebileceğini tartışıyoruz.

Anahtar sözcükler: *Otomatik plan düzenlemesi; hesaplamalı mimarlık; hesaplamalı tasarım; tasarımda evrimsel hesaplamalar; evrimsel tasarım.*

Department of Architecture, İstanbul Technical University Faculty of Architecture, İstanbul, Turkey.

Article arrival date: October 27, 2017 - Accepted for publication: September 14, 2018

Correspondence: Nizam Onur SÖNMEZ. **e-mail:** onursonmezn@yahoo.com

Introduction and Rationale For the Study

The Two Types of Inherent Complexities in Design Problems

In contrast to the well-defined problems, the problems that the designers tackle often exhibit characteristics that are referred to as ill-defined, ill-structured,[1] open-ended and even as wicked.[2] This contrast is important, because, as we move away from well-defined domains like chess and puzzle-solving, and get close to more open-ended domains like planning and design, an increase in the complexity of the problems compel the problem-solving behavior to change in a qualitative sense.[3]

In the case of architectural design, the products of a design process are complex entities, involving systems, subsystems, parts, sections, and functions that amount to an enormous variety of materials, details, and systems. This situation can be referred to as the "inherent complexity" of design.

A second kind of complexity concerns the negotiated aspects of design. At any design situation there will always be competing viewpoints, which have to be settled through negotiation. This issue has been among the reasons why design has been characterized as "wicked".[4] This type of complexity is related to the essentially undefined aspects of design, because it appears within a particular situation and it can only be solved by dispute or negotiation. This property of design situations may be referred to as the "inherent undefinedness" of design.

As an example of such complex problems, the Architectural Layout Design Problem (ALDP),[5] which is one of the primary tasks in architectural design, is concerned with the topological and geometrical assignment of activities to space such that a set of architectural criteria are met and some objectives are optimized. In its architectural variant, the layout problem is an extremely complicated problem, which is in connection with almost all aspects of a building's design. According to Russel and Norvig's classification scheme,[6] the task environment of a real world ALD Problem could be claimed to be partially observable, multi-agent, stochastic, sequential, dynamic, continuous, and unknown, i.e., the hardest case.

To reduce this complexity, many psychological and spatial aspects of the problem are traditionally omitted in –mainly engineering oriented- layout problem definitions, and the problem is mostly handled only in terms of efficiency, effective use of spaces, or cost, which is by no means a sufficient list for architectural aims. This is one of the reasons why, despite more than 40 years of effort, artificial architectural layout generation systems have not reached competence levels that are comparable to human designers'. There is no simple answer to the difficulties of ALDP; however, dynamic problem solving strategies appear as one of the requirements.[7]

As a problem-solving technique, Evolutionary Computing (EC) tries to approach optimal values closer and closer through the migration of a species of solution candidates within a complex search space. There has been a wide range of attempts to utilize EC for design and arts; however, most of these studies depend upon fixed problem definitions and do not utilize dynamic strategies. As examples of early studies, John Gero's research group studied a diverse array of tasks through EC. They experimented with space layout topologies, combination of Shape Grammars with evolutionary approaches, and evolving linear plan units as design genes within a two-phased hierarchical evolution.[8] Rosenman studied interactive evolution for floor plan generation[9] and Rosenman and Saunders experimented with a self-regulatory hierarchical co-evolution model for design.[10] These studies operated within highly constrained, simplified, and isolated sub-domains of architecture and mostly followed optimization approaches using just a few objectives, such as circulation costs calculated through pre-given adjacency matrices.

Likewise, the more developed applications appeared within rather well-defined sub-problems of architecture. A series of experimentations has been carried out by Caldas, Norford, and Rocha,[11] which use EC for the aim of integral building envelope design and performance optimization. Again, with a performance oriented design approach, Turrin, Buelow, and Stouffs developed an application to combine parametric modeling and EC.[12] Such studies resulted in a proliferation of performance oriented approaches to design generation, whose unifying characteristic is to assume a simplified, performance-oriented procedure, which render known optimization techniques directly applicable.

In brief, EC has been mostly used through static problem definitions, which do not respond well to the essentially vague, highly contextual, and consequently, highly dynamic manner of design processes.[13] Responding to such

[1] Simon, 1973.
[2] Rittel and Webber, 1973.
[3] Lawson, 2004, p. 19.
[4] Rittel and Webber, 1973; Buchanan, 1992.
[5] The abbreviations used in this paper are, Evolutionary Computing (EC), Evolutionary Algorithm (EA), Vector-Evaluated Genetic Algorithm (VEGA), Interleaved Evolutionary Algorithm (IEA), Non-dominated Sorting Genetic Algorithm 2 (NSGA2), Architectural Layout Design Problem (ALDP), Design Unit (DU), and Design-of-Experiments (DoE).
[6] Russell and Norvig, 2010, p.41-4.
[7] Sönmez and Erdem, 2014.
[8] Gero, Louis, and Kundu, 1994; Gero and Schnier, 1995; Damski and Gero, 1997; Gero and Kazakov, 1998; Jo and Gero, 1998.
[9] Rosenman, 1997.
[10] Rosenman and Saunders, 2003.
[11] Caldas and Rocha 2001; Caldas and Norford, 2002, 2003; Caldas, 2003, 2005, 2006, 2008.
[12] Turrin et al., 2011; Turrin, von Buelow, and Stouffs, 2011.
[13] A broad literature review and an in-depth discussion can be found in Sönmez and Erdem, 2014 and Sönmez, 2015a; 2015b.

characteristics of design situations through EC requires the development of design-oriented, dynamic Evolutionary Algorithms (EAs). Indeed, a recent proliferation of design-oriented optimization approaches[14] can be interpreted as pointing to a practical need for customized methods for specific design tasks. In many practical examples, customized variants and combinations of well-known approaches handle a specific task more efficiently. Exemplifying such a trend, Janssen[15] presents a parallel and distributed EA to handle high computational demands, Raphael[16] develops a novel multi-objective EA called RR-pareto, and in a series of papers, Rodrigues, Gaspar, and Gomes[17] present a hybrid evolutionary technique, which couples Evolutionary Strategies with Stochastic Hill Climbing.

Likewise, this study presents a novel multi-objective Evolutionary Algorithm (EA), i.e., the Interleaved EA (IEA). The IEA follows the above-mentioned tendency, as an initial step towards dynamic EAs for design. It has been developed with respect to an analogy with how a human designer works, who divides her design problems into smaller parts, i.e., focuses temporarily on a limited set of aspects to make complex problems more manageable.[18] In her dynamic decomposition strategy, the designer improves on her temporary problems through a pairwise integration strategy and moves forward to improve and reintegrate another set of aspects.[19] In an analogous manner, the IEA temporarily focuses on modules of objectives, although the integration is holistic, rather than pairwise. The key to holistic integration is to conserve the quality of the overall level and not to degrade any aspect too drastically while improving another, which is ensured by a multi-objective population selection stage.

In IEA, the ability to separate the operators and settings for each of the objectives gives the algorithm a modular structure. When it is suspected that the separate objectives may require different mutation and selection operators and could respond to different evolutionary settings (mutation and crossover numbers) and parameters (roulette parameters, mutation ratios, step sizes, etc.) the separation of operators, parameters, and settings may be beneficial. This separation also enables the user to adjust the objectives in terms of preferences through mutation and crossover settings.

Related Approaches and the IEA

IEA is a variant of the "criterion-based" methods for multi-objective optimization, which switch between the objectives during selection phases.[20] In criterion-based methods, at each selection stage, potentially different objectives decide which members of the population will be selected. Sometimes a probability is assigned to each objective, which determines whether the objective will be the sorting criterion in the next selection step. In the "Vector-Evaluated Genetic Algorithm" (VEGA) approach, selections are carried out according to each objective in turn. Offspring are then mixed, regardless of which objective dictated their selection.[21] In the "lexicographic" variant, the objectives are assigned priorities before optimization and the objective with the highest priority is used first when comparing individuals in a single-objective manner.[22]

In IEA, however, one of the objectives leads the evolution until its fitness progression stagnates, in the sense that the settings and fitness values of this objective is used for some of the evolutionary decisions. We call this the leading objective approach (Fig. 1). Note that a minimum number of generations is assigned to each new leading objective in the beginning of its lead (in practice, 8 to 32 generations), whether it stagnates or not. Thus, the IEA is differentiated from the earlier approaches with a dedicated period for the domination of each objective, whose duration is dependent on the feedback from the progression of the fitness values. This modification enables a set of dynamic methods, as will be discussed below.

In IEA, the new population is selected with a rank-based selection operator, considering all of the objectives simultaneously, which makes it essentially a multi-objective algorithm. However, variation selections and operations can be carried out according to the leading objective. In this way, separate mutation and selection operators and settings can be specified for each of the objectives within an overall task, which would be the same for all objectives in a regular multi-objective EA. Thus the specificities of the Interleaved EA are:

1. Letting each objective govern the process until its fitness improvement stagnates (whenever this occurs, the lead is given to another objective).
2. Enabling the use of separate operators, parameters, and settings for each objective.

In addition to its inherent adaptivity, the leading objective approach brings forth a potential to dynamically decompose a process through action packages that correspond to the process objectives. The rationale for this improvement will be discussed below.

Experimental setup for the Architectural Layout Design Problem

ALDP Representation

Because our ultimate aim is to tackle such complex

[14] Machairas, Tsangrassoulis, and Axarli, 2014.
[15] Janssen, 2009.
[16] Raphael, 2014.
[17] Rodrigues, Gaspar, and Gomes, 2013a; 2013b; 2013c.
[18] Akin, 2001; 2009.
[19] Akin, 2001.
[20] Zitzler, Laumanns, and Bleuler, 2004.
[21] Back, Fogel, and Michalewicz, 2000, p. 30-1.
[22] Back, Fogel, and Michalewicz, 2000, p. 29-30.

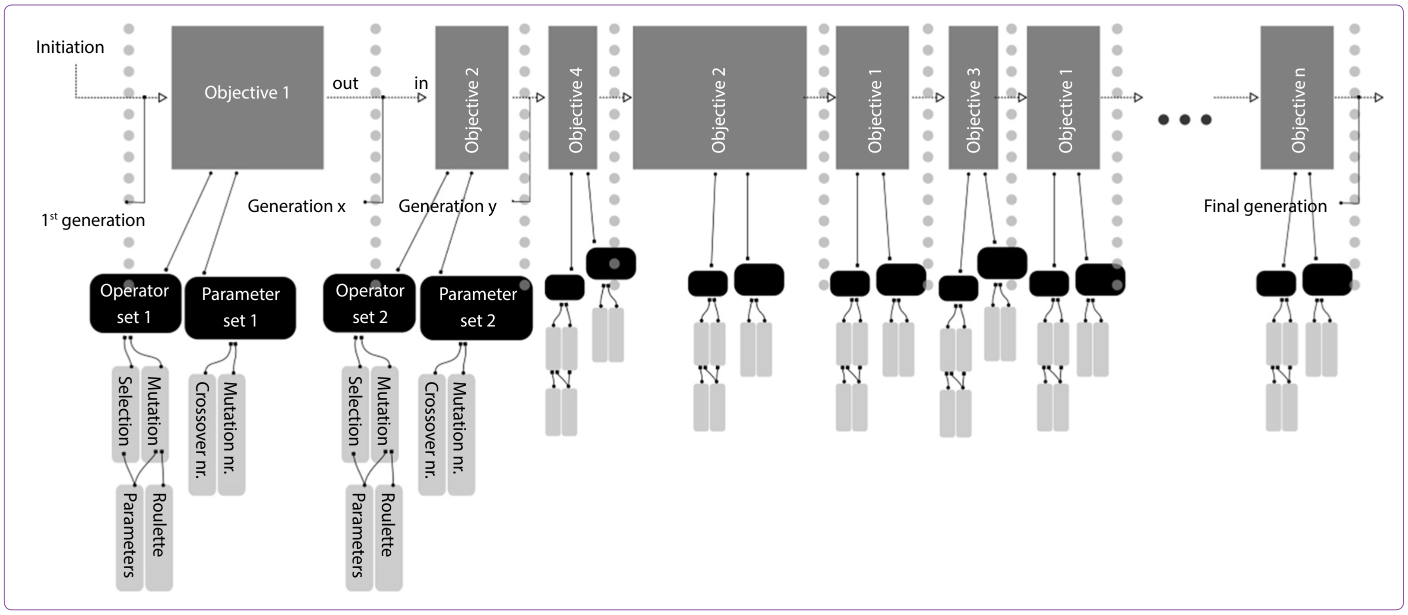

Figure 1. Schematic illustration of an Interleaved EA process.

problems in practice, instead of the generic problem types that are usually used for benchmarking in the EA literature, the IEA is examined in terms of the Architectural Layout Design Problem (ALDP).

Within the limits of this paper,[23] we define our experimental layout task as adequately populating a series of plan borders with a fixed set of arbitrary polygonal Design Units (DUs), whose forms, functions, and dimensions are determined and fixed at the outset (Fig. 2). The representation is flexible to enable overlapping between DUs and outer borders. Each DU has both an inner and an outer boundary. The inner boundary is used for measuring and penalizing overlap situations, while the outer boundary may be used for detecting and rewarding neighborhoods. In the initiation phase, fixed DUs are initiated within their pre-specified position, the remaining DUs are randomly placed within the bounding box of a given floor outline.

Evolutionary Specifications

The genotype of a candidate layout comprises a single list of fixed and movable DUs. The sequence of the DUs in this list is the same for all candidates. For each DU, DU type, center coordinates, inner and outer polygon coordinates, bounding box, and related geometric information are stored.

An N-point Crossover operator is used as default. For variation selections, the options are Uniform and Tournament (size 2) operators, while a Rank-based selection

[23] Test cases involve simplifications and do not correspond to a real world usage. This paper presents only the aspects that are relevant to the presented set of tests; a small and efficient set is chosen from available operators. For other aspects, details, and implementations of ALDP and IEA, please see Sönmez, 2015a; 2015b.

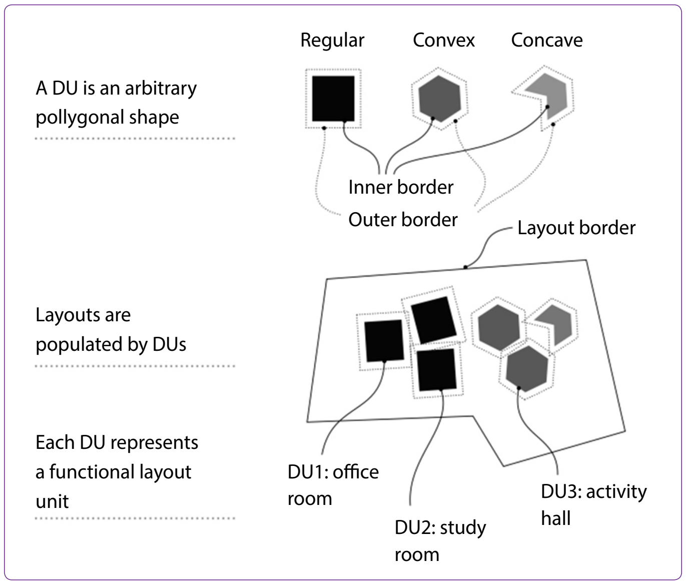
A DU is an arbitrary pollygonal shape
Regular
Convex
Concave
Inner border
Outer border
Layout border
Layouts are populated by DUs
Each DU represents a functional layout unit
DU1: office room
DU2: study room
DU3: activity hall

Figure 2. Basics of ALD Problem representation.

operator is used for new population selection. In Rank-based selection, all objectives are taken into account and the minimum rank number of an individual determines its selection probability as follows: For each objective, each candidate has a separate rank amongst all candidates. The individuals are listed according to their minimum-ranks. Starting from the highest available minimum-rank, a desired number of individuals are selected. If a series of individuals occupy the same minimum-rank level, rank values of each individual are summed up, and the individuals are ordered in terms of the resulting values. This approach tends to favor the candidates, which are reasonably fine on all objectives simultaneously, while eliminating candi-

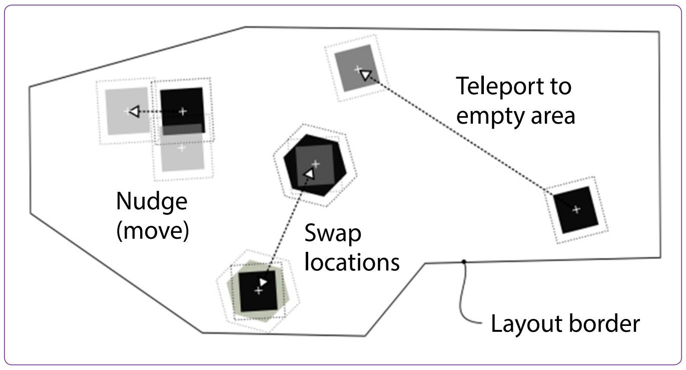

Figure 3. Mutation operators.

dates, which occupy low ranks on an objective, regardless of their being high rank individuals on others.

Three different mutation operators are used, i.e., "Swap", "Nudge", and "Teleport" (Fig. 3). Swap mutation exchanges the center locations of two DUs. Nudge mutation translates a DU in X or Y direction. Teleport mutation carries the center of a DU to an empty space within layout boundaries. Each objective has a mutation roulette, which includes different probabilities for each mutation operator. During the process, for each mutation candidate, a mutation operator is probabilistically drawn from this roulette. The operators, in turn, function stochastically through a set of parameters (nudge step σ and swap rates), which are also different for each objective.

Evaluation Procedures For the Objectives

Four objectives have been used for the application, i.e., "Overlap", "Trivial Hole", "Neighbor", and "Neighbor Cell" (Fig. 4). The aim of the Overlap objective is to keep

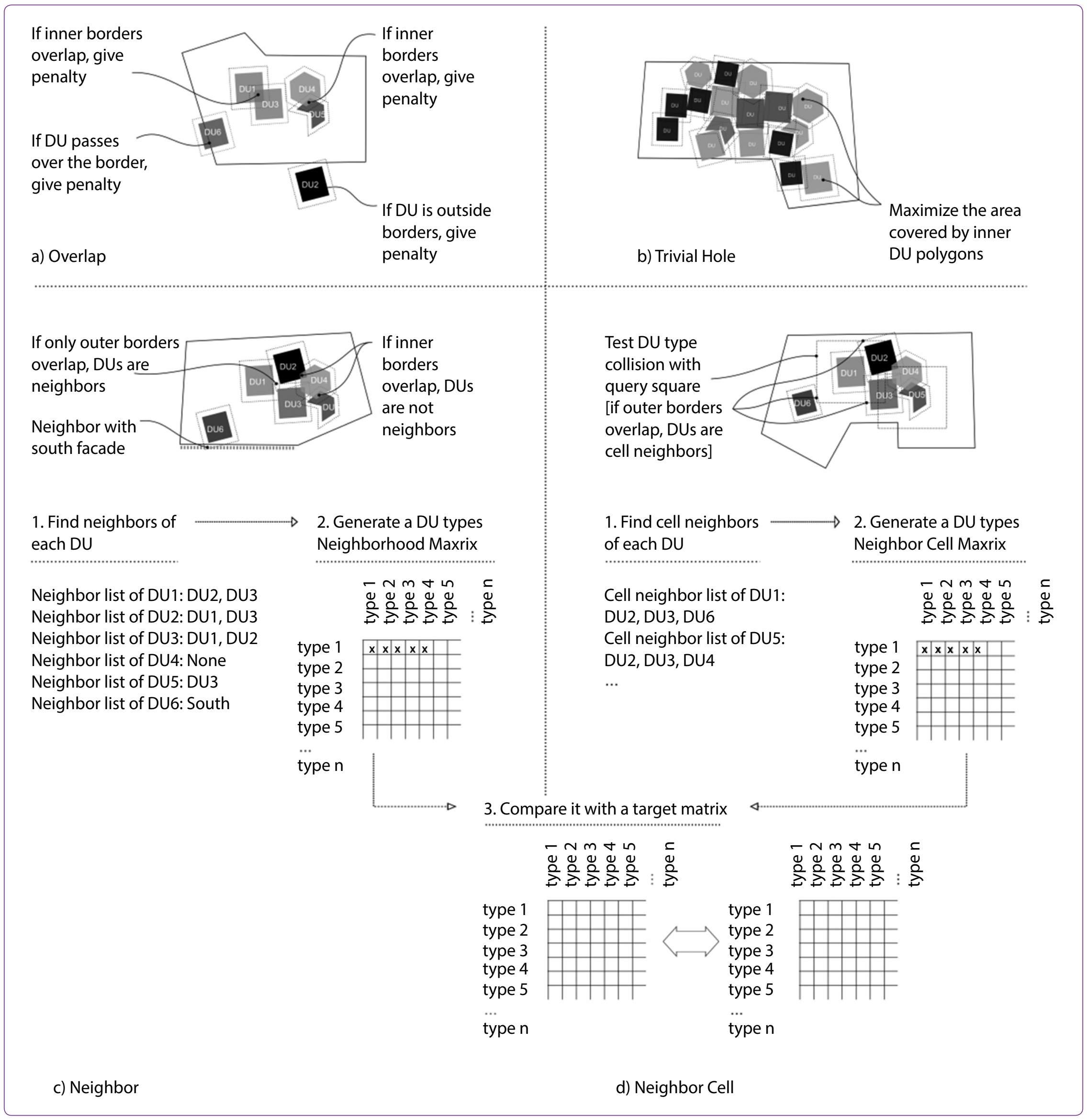
If inner borders overlap, give penalty
If inner borders overlap, give penalty
If DU passes over the border, give penalty
If DU is outside borders, give penalty
a) Overlap
b) Trivial Hole
Maximize the area covered by inner DU polygons
If only outer borders overlap, DUs are neighbors
If inner borders overlap, DUs are not neighbors
Neighbor with south facade
Test DU type collision with query square [if outer borders overlap, DUs are cell neighbors]
1. Find neighbors of each DU
2. Generate a DU types Neighborhood Maxrix
1. Find cell neighbors of each DU
2. Generate a DU types Neighbor Cell Maxrix
Neighbor list of DU1: DU2, DU3
Neighbor list of DU2: DU1, DU3
Neighbor list of DU3: DU1, DU2
Neighbor list of DU4: None
Neighbor list of DU5: DU3
Neighbor list of DU6: South
Cell neighbor list of DU1: DU2, DU3, DU6
Cell neighbor list of DU5: DU2, DU3, DU4
...
type 1 type 2 type 3 type 4 type 5 ... type n
3. Compare it with a target matrix
c) Neighbor
d) Neighbor Cell

Figure 4. Calculations for the four objectives.

the DUs within plan limits while preventing DU overlap. The procedure calculates a weighted sum of the penalties given to layout border violation and to inner DU boundary overlaps. First, all DUs are checked for whether their inner boundaries violate layout border. If a DU is partially violating the border, a penalty is given according to the ratio of the violating part. If a DU is totally out of the boundaries, a penalty is given, proportional to the square of the DUs distance from the border. Secondly, all DUs are checked for overlapping other DUs (inner boundary). If overlap occurs, the overlapping area is divided by the smaller DU's area, and a penalty is given accordingly. Finally, total layout border penalties and DU overlap penalties are combined into a single fitness value by a weighted sum.

The Trivial Hole objective measures the ratio of the occupied area within plan borders, as maximizing this value

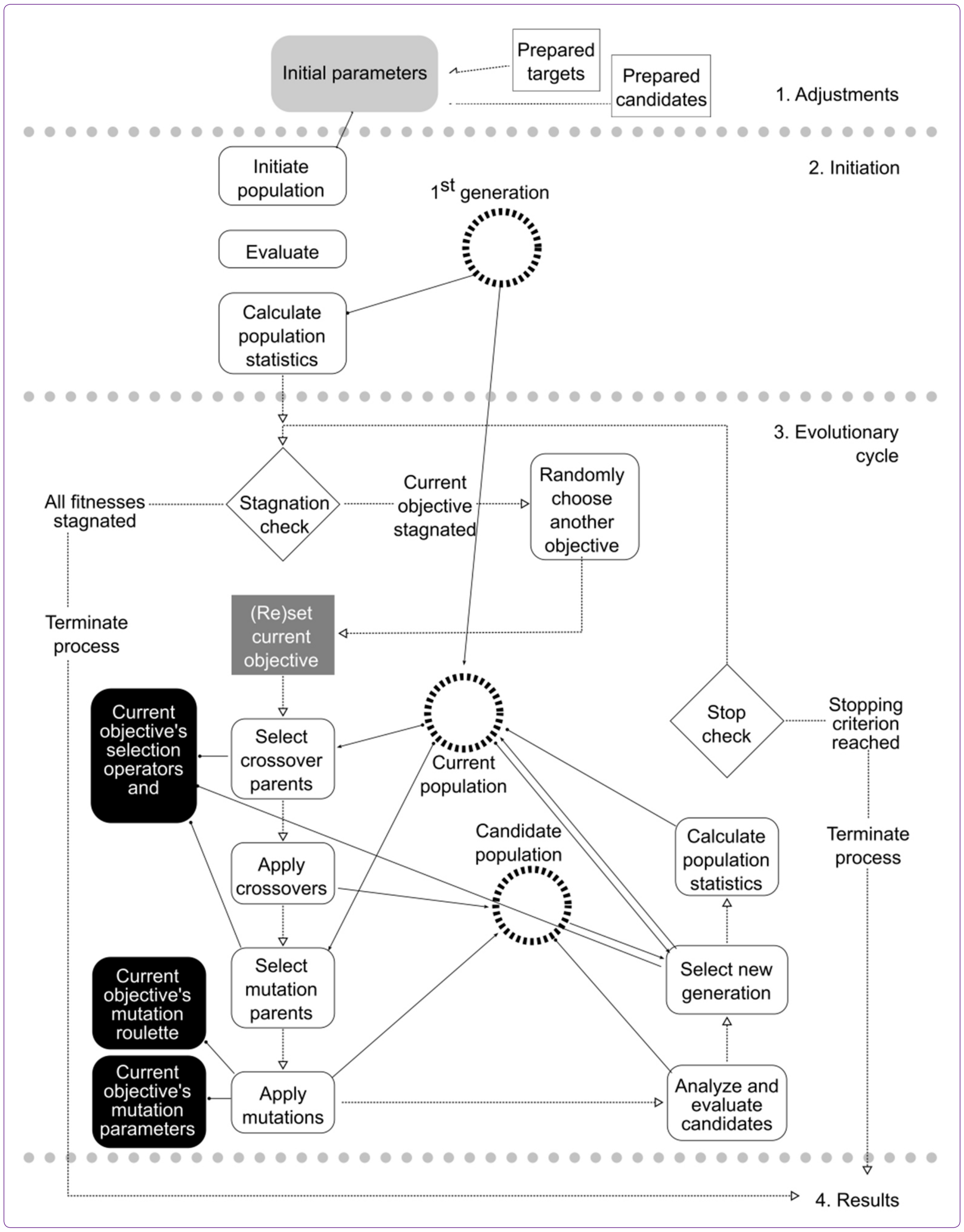

Figure 5. Interleaved EA, evolutionary process.

amounts to the minimizing of DU overlap and layout border violation; and implicitly, minimizing the empty area within borders.

The Neighbor objective tries to maximize the similarity of a candidate's neighborhood distribution to a target layout's. As such, it is thought as an alternative to adjacency matrices, which are usually static and are prepared by hand. This method enables the dynamic definition of a similar matrix through example buildings. A target is prepared as follows: The DUs of example layouts are vectorized with a color scheme. The direction that a DU is neighboring is also given with this diagram. Target preparation amounts to the collecting of the frequencies of neighborhoods between DU types within a neighborhood matrix. The rows and columns of the matrix are symmetrical and their sequence is fixed for all applications. The frequencies of neighborhood types are normalized by the maximum frequency. This way, the matrix becomes a neighborhood pattern, which can be compared with other examples. For candidate evaluation, neighborhoods of DUs are extracted from legitimate overlaps, i.e., overlapping of outer boundaries while inner boundaries remain non-violated. Outer DU boundary collision with the direction lines are used for the detection of direction neighborhoods. When the neighbors for each DU type are determined, this information is again converted into a normalized neighborhood pattern matrix. For fitness calculation, the absolute values of the differences of corresponding target and candidate half-matrix cells are summed up. The resulting value is negated to treat this error value as positive fitness.

The Neighbor Cell objective is similar in its aims to the Neighbor objective; however, the detection method for the spatial adjacencies is different. A fixed sized square query cell is placed over each DU's center. Types of all the DUs that collide with a DU's query cell are added to the neighbor list of that DU. For candidate evaluation, a target's matrix, which is similarly prepared, is compared with the candidate's. As with the Neighbor objective, Neighbor Cell objective calculates and negates the difference between two matrix patterns, while adjacencies to directions are not taken into account. It should be noted that, separate targets may be used for each objective.

Evolutionary Processes For IEA and NSGA2

The basic IEA process is as follows: In the first step, the candidate layouts are initiated, the default objective is set as the leading objective, and all candidates are evaluated for all objectives. The candidates for crossover and mutation are selected using the leading objective's selection operators and parameters. Then the variation operations are carried out. After a pre-specified number of generations (t), which determines the minimum number of generations for each leading objective, stagnation checks are carried out. A slope-based method is used for stagnation control: The fitness progression graph of an objective is fitted with a line through linear regression for N last generations. The slope of this line shows the recent progression of the process for that objective and the stagnation threshold denotes a minimum slope. If the leading objective's fitness improvement is found stagnated for the last t generations, one of the objectives is randomly drawn from the list of objectives as the next leading objective. This method may also be used as a stopping criterion (if all objectives stagnate for a period of generations, stop evolution). However, in the following test cases, stopping criterion is a fixed number of generated offspring.

A simple approach is used to preserve diversity:[24] If the fitness value combination of a new offspring already exists in the population, it is not inserted into the new population. Although the population number is kept fixed for the test cases, a simple elitism scheme adds the best candidate for each of the objectives to the next generation, if not already selected. This procedure can increase the population number, at most for the number of objectives.

In the following test cases, the IEA will be compared to the "Non-dominated Sorting Genetic Algorithm 2" (NSGA2).[25] In NSGA2, first, chromosomes are sorted and put into fronts according to Pareto dominance. Within a Pareto front, the chromosomes are ranked with regard to the distance between the solutions. Solutions that are far away from other solutions are given a higher preference for selection. This is done in order to obtain a diverse solution set. Therefore, the two essential differences of the two EAs are, (1) while the IEA has a simple diversity strategy, the functioning of the NSGA2 essentially depends on the preservation of diversity; (2) while NSGA2 searches for Pareto non-dominance, IEA uses a rank-based population selection strategy, not explicitly maintaining Pareto non-dominance.

Additionally, a regular rank-based version of IEA is generated by simply using a single parameter and operator combination for all objectives. The algorithm is the same with IEA in other respects. This version is used for assessing the effect of the leading objective approach.

Because IEA and its rank-based version use the above-mentioned elitism scheme, the length of the evolutionary processes for the three EAs are measured, fixed, and equalized over the total number of generated candidates, instead of the generation number. This is also a better estimate for the required amount of computation, as each of the candidates has to be evaluated once, which is the most time consuming aspect of our problem.

[24] From, Michalewicz and Schmidt, 2007.

[25] The implementation is based on Deb et al. 2002.

Table 1. Library buildings used in the preparation of targets (For all sources, last access, October, 2015)

Case	Name of Library	Architect	More Info
1.	Actur Library, 2008	Carroquino Finner Arquitectos.	http://www.archdaily.com/23128/actur-library-carroquino-finner-arquitectos/
2.	Central Library, Universidad Catolica del Norte, 2002	Marsino Arquitectos Asociados	http://www.archdaily.com/2742/central-library-universidad-catolica-del-norte-marsino-arquitectos-asociados
3.	Santa Monica Public Library, 2006	Moore Ruble Yudell Architects	http://www.moorerubleyudell.com/projects/santa-monica-public-library

Test Cases

Three cases of ALD Problems that are simplified versions of real buildings (Table 1) with differing degrees of complexity have been chosen for the test series (Figs. 6-8). Each case is determined by a candidate scheme, which comprises layout borders, fixed DUs and a set of movable

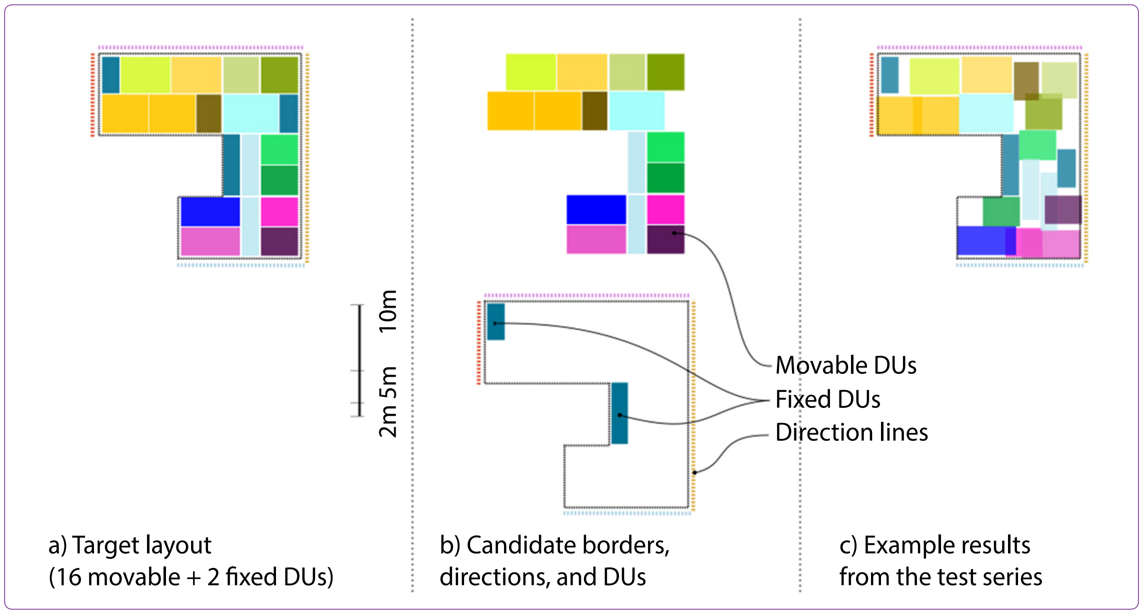

Figure 6. Test case 1.

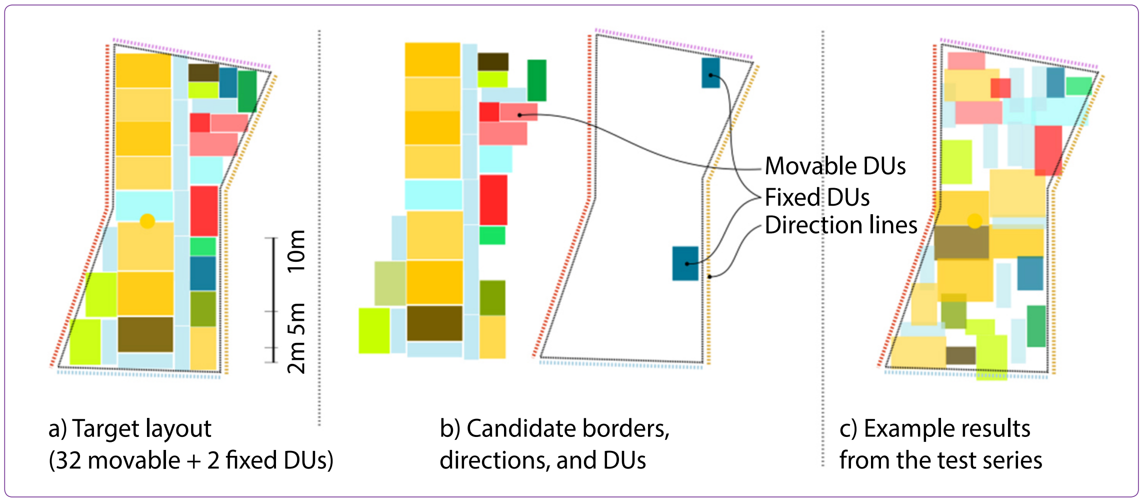

Figure 7. Test case 2.

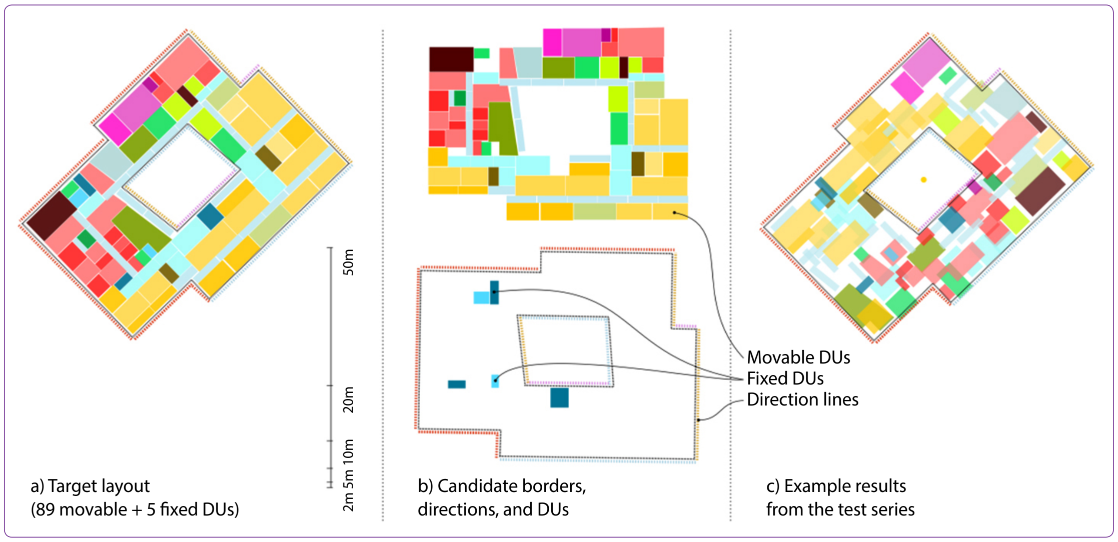

Figure 8. Test case 3.

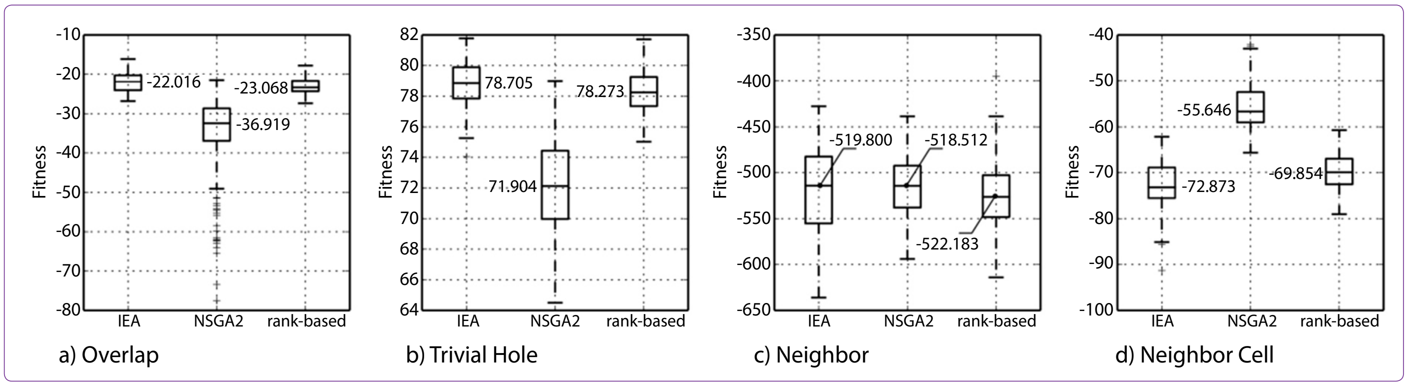

Figure 9. Test Case 1, distribution of 100x5 high-ranking results (IEA, NSGA2, Rank-based).

DUs. This scheme is used for the initiation of all candidate layouts. The aim is to arrange the given movable DUs within each layout for generating patching drafts. The target for each case is the original layout that is used in the preparation of the candidate schemes. For each test case, targets and candidates have the same list of DUs and layout borders. This is not meaningful for real world usage; yet, it is informative for testing aims. Note that, except for the simplest and most constrained cases, attaining of the real target is not expected.

Parameter Finding Process

For a proper comparison between the IEA, its rank-based version, and NSGA2, the EAs have to be run with their best parameter combinations. However, in each of the EAs, a high number of parameters can have an effect on the quality of the results, and it is not practical to find the best parameter combinations in a single trial set, as there would be an exponential number of combinations. Therefore, a combined ‘Design-of-Experiments’ (DoE) and ‘Racing’ procedure was devised.

As a first step, the number of varied parameters was reduced. Secondly, a small set of discrete values was determined for each parameter (see Table 2). Thirdly, through a series of consecutive racing processes, initially the most important parameters were found and fixed (selection operators, mutation and crossover ratios), and then, gradually, roulette and mutation parameters were progressively fine-tuned. Because the process is multi-objective, a pare-to-racing procedure was implemented as follows:[26]

- Generate a set of combinations, each of which defines an EA instance,
- Run each of these EA instances for n turns (5 or 10 turns),
- After n turns, at each new turn,
    - o Compare each combination with each of the others as follows:
        - For each combination,
        - For each of the objectives,

[26] Based on, Yuan and Gallagher, 2007 and Zhang, Georgiopoulos, and Anagnostopoulos, 2013.

Table 2. Parameters and values for test cases

	Rank-based	NSGA2		IEA
Population	1000	1000		1000
Max. num. of cands.	~300000	~300000		~300000
Crossover selection	uniform	uniform		uniform
Crossover ratio	1	1		0,02
Mutation selection	uniform	uniform		uniform
Mutation ratio	0,7	0,7	**Mutation ratios**	
			Trivial Hole	0,01
			Overlap	0,02
			Neighbor	0,001
			Neighbor Cell	0,001
Roulette parameters				
Nudge rate	3	2	Overlap Nudge rate	3
Teleport rate	2	1	Overlap Teleport rate	1
Swap rate	2	1	Overlap Swap rate	1
Nudge step (σ)	2	2	Overlap Nudge step (σ)	0,8
			Trivial Hole Nudge rate	3
			Trivial Hole Teleport rate	1
			Trivial Hole Swap rate	1
			Trivial Hole Nudge step (σ)	0,8
			Neighbor Nudge rate	2
			Neighbor Teleport rate	1
			Neighbor Swap rate	3
			Neighbor Nudge step (σ)	1,5
			Neighbor Cell Nudge rate	3
			Neighbor Cell Teleport rate	1
			Neighbor Cell Swap rate	2
			Neighbor Cell Nudge step (σ)	2

- o Get the distribution of the previous fitness values,
- o Compare the fitness value distributions of the two combinations using Welch's t test,[27]
- o If the two combinations are significantly different (i.e., p < %5 or %10):
    - Find the fitness value averages of each combination.
    - Increase the counter of the worse combination by 1.
    - After the two combinations are compared for all objectives, if one of the combinations is found to be worse on all objectives, it is dominated by the other. Therefore, it is left out of the population for the next turn.

In this way, the amount of necessary computation is decreased at each turn and the process resembles a race, which continues until either only one candidate remains or a pre-specified number of turns is reached (50 or 100). If there is a set of remaining combinations, the best individual is selected amongst these, using the same rank-based selection operator used for IEA. The value combination of this instance is fixed for the next level of parameter finding.

The best parameter/value combinations are found for each of the EA types, using the simplest test case (Fig. 6). Table 2 presents the parameters and found value combinations for each of the EAs. The varied parameters are crossover rate, mutation rate, roulette rates (probabilities) for each of the mutation operators and σ value for nudge mutation. For the Interleaved EA, these parameters are different for each of the objectives. As can be observed, different best value combinations have been found for different objectives in IEA, and also amongst the different EA types.

Tests and Results

A set of 100 trials were run for each of the EAs and for each of the test cases, which results in a matrix of 9 trial

[27] Python Scipy's "Welch's t test" implementation is used for the trials.

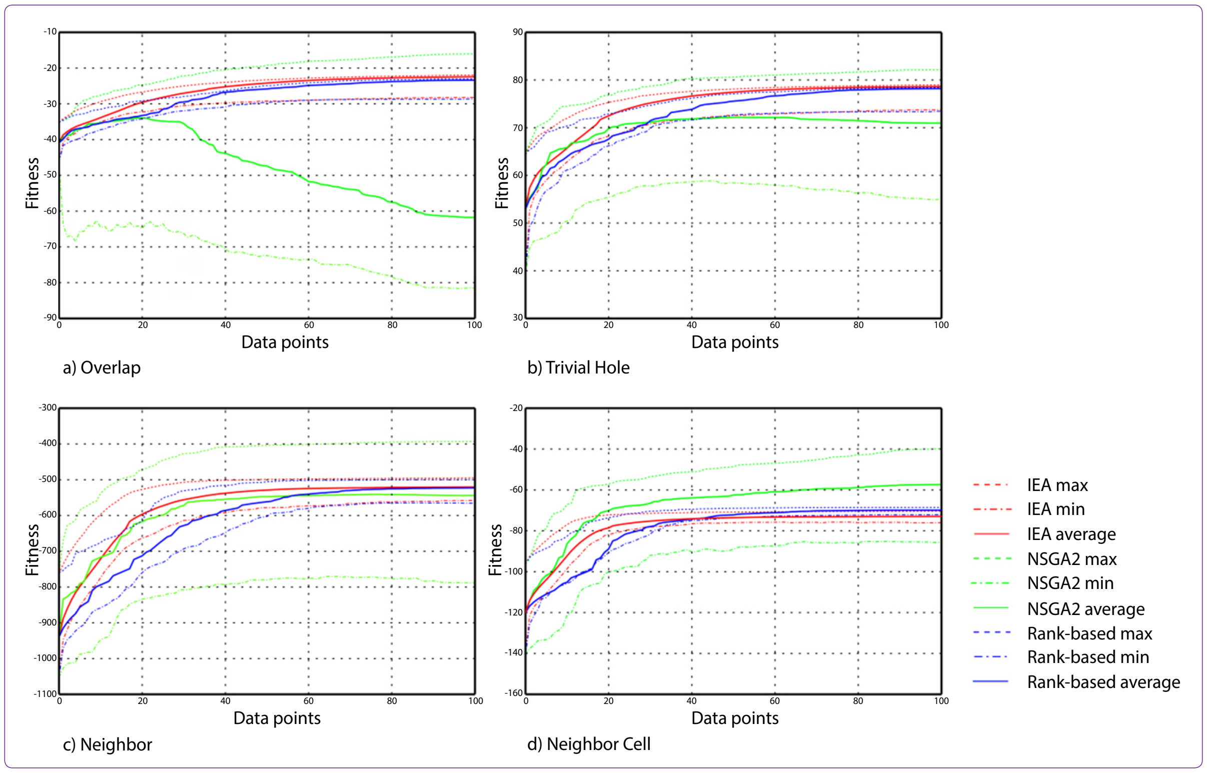

Figure 10. Test Case 1, Process graphs for IEA, NSGA2, and Rank-based version.

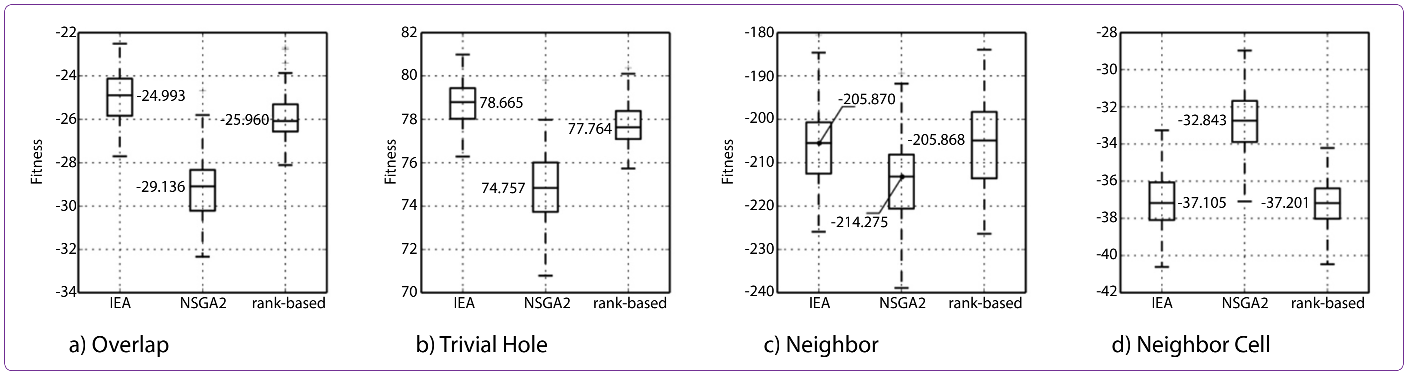

Figure 11. Test Case 2, distribution of 100x5 high-ranking results (IEA, NSGA2, Rank-based).

sets. The two main points to explicate with these trials is (1) to show that the core innovation (i.e., the leading objective principle) does not bring a disadvantage (hence the comparison with the regular rank-based case), and (2) to show that the overall performance is comparable to the state-of-the-art methods, which is the motivation behind the comparison with NSGA2.

Two different types of visualizations are prepared for the inspection of the results. A set of graphics compare mean fitness progression graphs for each EA type (Figs. 10, 12, 14) where minimum, maximum, and average fitness values are given. Each of these lines represents the mean of 100 trials. For a design problem, the results have to be fine on all objectives simultaneously, as a product that is considerably bad for one objective is practically useless, even if it is optimal for another objective. Thus, in the process graphs, the progression of the average fitness values of populations has a practical significance. However, because of its diversity preservation mechanism, NSGA2's average fitness values tend to appear lower than the other EAs. Therefore, box plots are presented as a second assessment method (Figs. 9, 11, 13), showing the fitness distribution of the 500 best results for each EA (5 best results from each of the 100 trials are selected using the same rank-based selection operator with the IEA). It should be noted that, for practical aims, this appears as a better assessment method, because these best results are what would be used in practice.

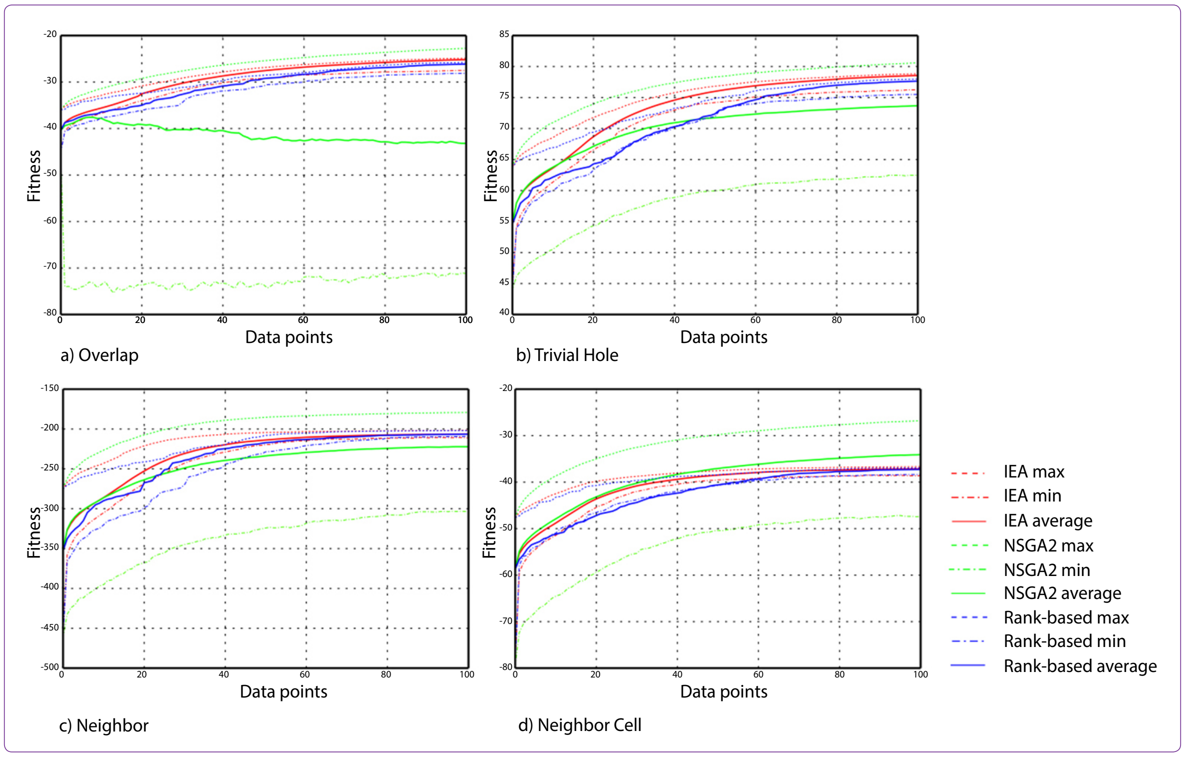

Figure 12. Test Case 2; Process graphs for IEA, NSGA2, and Rank-based.

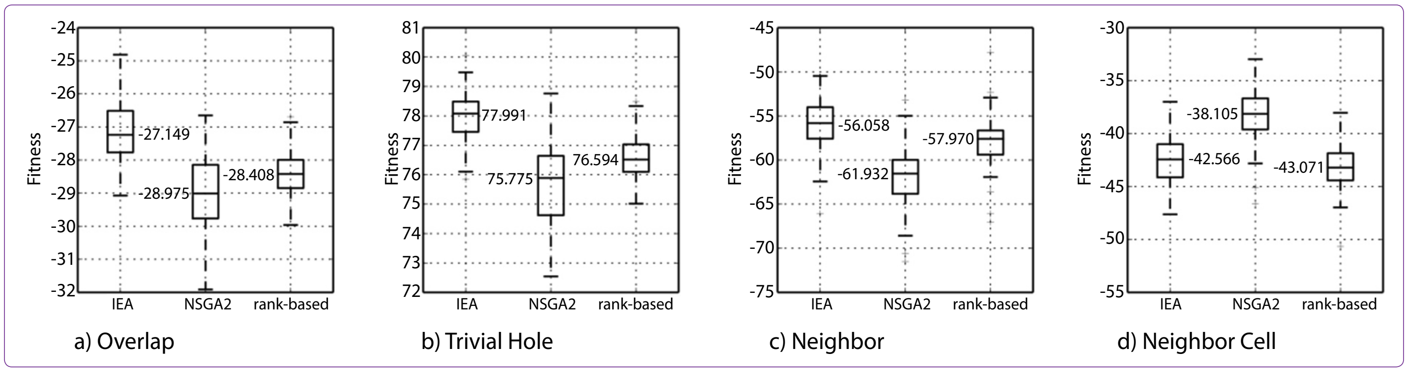

Figure 13. Test Case 3, distribution of 100x5 high-ranking results (IEA, NSGA2, Rank-based).

Comparing the IEA to the rank-based version, for the simplest case (Figs. 9, 10) the final fitness levels do not deviate considerably; however, as the problem gets more complicated (i.e., case 3, Figs. 13, 14) the differences between the two approaches become more salient, in both attained fitness levels and the speed in arriving high levels of fitness, where IEA appears advantageous. Faster convergence can be interpreted to express the adaptive character of the leading objective principle. As architectural fitness calculations can become time consuming, and considering that these applications will be used on regular desktop computers, faster fitness improvement has a practical advantage. Thus using dedicated parameter sets for each of the objectives can generate a practical advantage over the traditional ‘same parameter set’ approach.

Comparing the functioning of the IEA with NSGA2, while NSGA2 consistently attains better maximum fitness levels (Figs. 10, 12, 14), from a practical perspective, this does not guarantee the usability of the proposals evolved by the NSGA2, in the sense that these are reasonably fine on all objectives. This is indicated by the distributions of the best usable proposals (Figs. 9, 11, 13). In practical terms, the IEA appears to yield more usable proposals for three of the objectives (i.e., Overlap, Trivial Hole, and Neighbor), while NSGA2 has dominance on the Neighbor Cell objective (Figs. 9, 11, 13).

The Neighbor and Neighbor Cell objectives measure for a similar characteristic, yet with different methods. It appears interesting to compare these two procedures to shed more light on the character of the IEA. The Neighbor

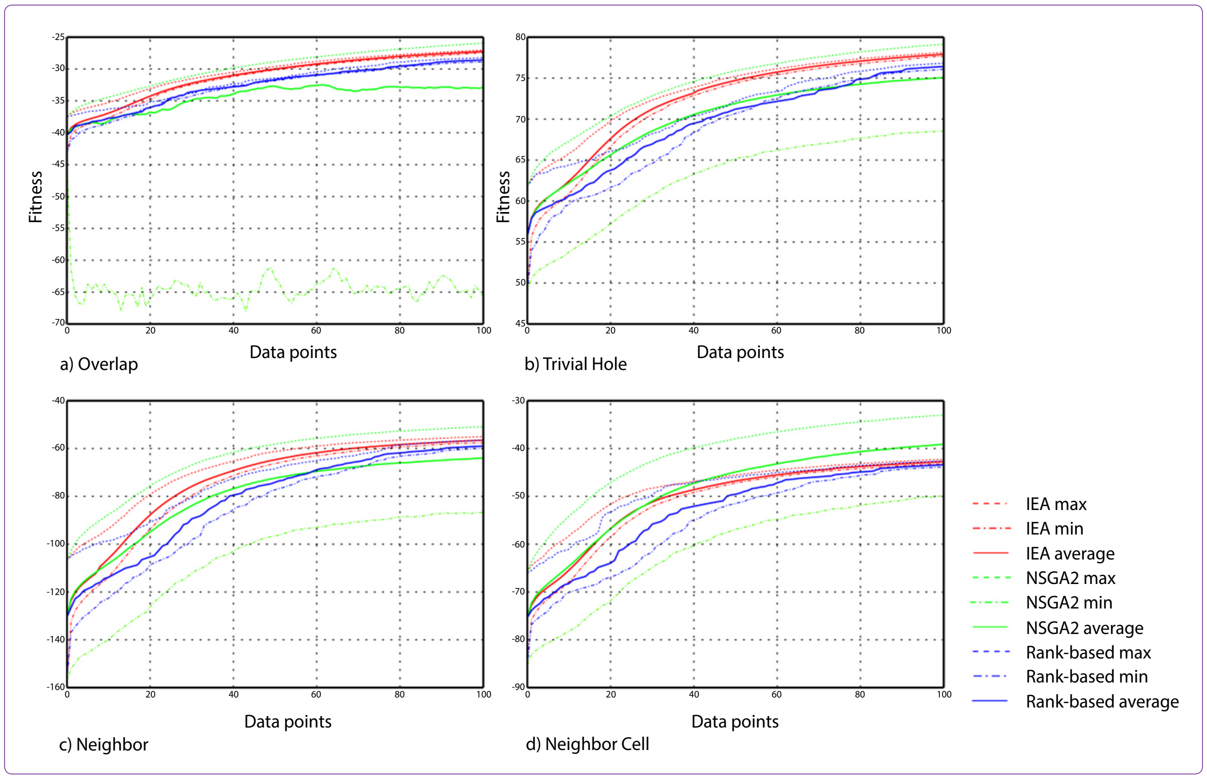

Figure 14. Test Case 3, Process graphs for IEA, NSGA2, and Rank-based.

Cell objective has a less discriminating and more permissive measurement of neighborhood, as its detection area is much larger than the Neighbor objective, which accepts only the area between outer and inner borders of a DU (Fig. 4). Likewise, the two other objectives, for which the IEA exhibits practical advantage, apply fine-grained measurements for their fitness calculations and they are rather discriminating. Compared to the other three objectives, the Neighbor Cell objective is less discriminating and allows more variety, which is consistent with the overall philosophy of NSGA2, which depends on the maintaining of variety. On the other hand, the rank-based population selection mechanism of the IEA favors convergence towards a smaller search region, eliminating outliers, in order to attain reasonable products, rather than variety.

Discussion

The above trials should be considered as providing insight for the IEA's specific functioning. The trials had several shortcomings that have to be indicated. The number and types of the objectives, the set of varied parameters, and the possible values for these parameters were reduced and this brought forward coarse-grained test cases. As a result, our DoE + Racing approach could only produce an approximation of the best value combinations. Additionally, the values were fine-tuned only for the simplest Actur case and used also for the more complex cases.

Besides the high computational load of the fitness evaluations, another reason in the above shortcomings was the combinatorial nature of the IEA, whose number of parameters increases linearly with the number of objectives. To alleviate this abundance of parameters, adaptive parameter tuning approaches have to be integrated to the IEA. However, as the parameter number of an EA increases, the effectivity of adaptivity schemes decreases. Although this indicates a crucial weakness of the IEA, in practice, the algorithm has been used in several cases satisfactorily with rather coarse values.[28] This is due to the undemanding task settings, which aimed at the development of crude draft designs, in the absence of human-level sensitive evaluation methods that could make high-quality proposal development a practical aim. This is also the reason for prioritizing speed and reasonable results.

In any case, it is not the practical performance gain that is the reason behind the IEA, but rather the future development potentials of the approach. The ability to separate the operators and settings for each of the objectives gives the IEA a modular structure. This modularity offers a method for the utilization of domain-specific knowledge for each sub-task, i.e., objective. What is aimed in such modularity is an evolutionary process, which would analyze the situation at each important phase, and then revise its operator set according to its evaluations. This would

[28] Sönmez, 2015a.

render the development process a dynamic and intelligent one as required. In the above test cases, the transition between objectives in IEA depends only on the feedback from fitness progressions. In principle, other analyses could also be used for feedback (e.g. on the details and statistics of the states and performances of the evolved populations). Note that this is related to the problem definition and available analysis tools, and not with the IEA itself, which already offers the potential for dynamism. Thus the IEA indicates where new intelligent technologies could be inserted; in other words, it is essentially open to further development through additional methods to determine when to follow which objective and with which operators, in which case the IEA would demonstrate its full potential as a truly dynamic EA.

Acknowledgements

The initial stage of this research was supported by TUBITAK (The Scientific and Technical Research Council of Turkey) and Istanbul Technical University Scientific Research and Development Support Program.

References

Akin, Ö. (2001) “Variants in Design Cognition”, in Eastman, C., Newstetter, W., and McCracken, M. (eds.), Design Knowing and Learning: Cognition in Design Education, 978-0-08-043868-9, Elsevier, http://doi.org/10.1016/B978-008043868-9/50006-1.

Akın, Ö. (2009) “Variants and Invariants of Design Cognition”, in McDonnell, J. and Lloyd, P. (Eds), About Designing: Analysing Design Meetings, CRC Press, pp. 171-192.

Back, T., Fogel, D.B., and Michalewicz, Z. (Eds.) (2000) Evolutionary Computation 2: Advanced Algorithms and Operators, IOP Publishing Ltd, Bristol and Philadelphia.

Buchanan, R. (1992) “Wicked problems in design thinking”, Design Issues, Vol. 8, No. 2, pp. 5-21.

Caldas, L.G. (2003) “Shape generation using pareto genetic algorithms”, CAADRIA 2003.

Caldas, L.G. (2005) “Three-dimensional shape generation of low-energy architecture solutions using Pareto GA’s”, Proceedings of ECAADE’05, Lisbon, September 21-24, 2005, pp. 647-654.

Caldas, L.G. (2006) “GENE_ARCH: An evolution-based generative design system for sustainable architecture”, I.F.C. Smith (Ed.), EG-ICE 2006, LNAI 4200, pp. 109 – 118, 2006.

Caldas, L.G. (2008) “Generation of energy-efficient architecture solutions applying GENE_ARCH: An evolution-based generative design system”, Advanced Engineering Informatics, Volume 22, Issue 1 (January 2008).

Caldas, L.G. and Norford, L.K. (2002) “A design optimization tool based on a genetic algorithm”, Automation in Construction, 11 (2002) 173 – 184.

Caldas, L.G. and Norford, L.K. (2003) “Genetic Algorithms for Optimization of Building Envelopes and the Design and Control of HVAC systems”, Journal of Solar Energy Engineering, August 2003, Vol. 125.

Caldas, L.G. and Rocha, J. (2001) “A generative design system applied to Siza’s school of architecture at Oporto”, In J.S. Gero, S. Chase and M. Rosenman (Eds) CAADRIA 2001, Key Centre of Design Computing and Cognition, University of Sydney, 2001, pp. 253-264.

Damski, J.C. and Gero, J.S. (1997) “An evolutionary approach to generating constraint-based space layout topologies”, In R. Junge (Ed.), CAADFutures 1997, Kluwer, Dordrecht. pp. 855-864.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002) “A fast and elitist multiobjective genetic algorithm: NSGA-II”, IEEE Transactions on Evolutionary Computation, 6(2), pp. 182–197.

Gero, J.S., Louis S., and Kundu, S. (1994) “Evolutionary learning of novel grammars for design improvement”, AI EDAM, 8(2):83-94.

Gero, J.S. and Schnier, T. (1995) “Evolving representations of design cases and their use in creative design”, Third International Conference on Computational Models of Creative Design.

Gero, J.S. and Kazakov, V.A. (1998) “Evolving design genes in space layout planning problems”, Artificial Intelligence in Engineering, 12 (1998) 163-176.

Janssen, P. H. (2009) “An evolutionary system for design exploration”, in Proceedings of the International Conference on Computer Aided Architectural Design Futures, Montréal, Canada17th-19th June, pp. 260–272.

Lawson, B. (2004) What Designers Know, Elsevier / Architectural Press, Amsterdam.

Machairas, V., Tsangrassoulis, A., & Axarli, K. (2014) “Algorithms for optimization of building design: A review”, Renewable and Sustainable Energy Reviews, 31, pp. 101–112, http://doi.org/10.1016/j.rser.2013.11.036

Michalewicz, Z. and Schmidt, M. (2007) “Parameter Control in Practice”, in Lobo, F.J., Lima, C.F., and Michalewicz, Z. (Eds.), Parameter Setting in Evolutionary Algorithms, Series: Studies in Computational Intelligence, Vol. 54, XII, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 277-294.

Raphael, B. (2014) “Multi-Criteria Decision Making for the Design of Building Facade”, in Computing in Civil and Building Engineering, ASCE, pp. 1650–1658.

Rittel, H. W. J. and Webber, M. M. (1973) “Dilemmas in a General Theory of Planning”, Policy Sciences, 4, pp. 155-169.

Rodrigues, E., Gaspar, A.R., and Gomes, Á. (2013a) “An approach to the multi-level space allocation problem in architecture using a hybrid evolutionary technique”, Automation in Construction, 35, pp. 482–498, doi:10.1016/j.autcon.2013.06.005

Rodrigues, E., Gaspar, A.R., and Gomes, Á., (2013b) “An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology”, Computer-Aided Design, 45, pp. 887–897, doi:10.1016/j.cad.2013.01.001

Rodrigues, E., Gaspar, A.R., and Gomes, Á. (2013c) “An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 2: Validation and performance tests”, Computer-Aided Design, 45, pp. 898–910, doi:10.1016/j.cad.2013.01.003

Rosenman, M.A. (1997) “The generation of form using an evolutionary approach”, [online] http://arch.usyd.edu.au (Accessed: September 2009).

Rosenman, M.A. and Saunders, R. (2003) “Self-regulatory hierarchical coevolution”, AI-EDAM, 2003, 17, 273 – 285.

Russell, S. J. and Norvig, P., (2010) Artificial Intelligence: A Modern Approach, Prentice Hall (third edition).

Simon, H. A. (1973) “The Structure of Ill Structured Problems”, Artificial Intelligence, 4 (3): pp. 181–201.

Sönmez, N.O., Erdem, A. (2014) “Design games: A conceptual framework for dynamic evolutionary design”, A | Z ITU Journal of the Faculty of Architecture, Vol. 11, No. 1, 03/2014.

Sönmez, N.O. (2015a) Evolutionary Design Assistants for Architecture, ABE, Architecture and the Built Environment, 5(3), pp. 1–284, http://doi.org/10.7480/abe.2015.3

Sönmez, N.O. (2015b) “Architectural Layout Evolution through Similarity-Based Evaluation”, International Journal of Architectural Computing (IJAC), Vol. 13, No. (3-4), 10/2015, pp. 271–298, http://doi.org/10.1260/1478-0771.13.3-4.271

Turrin, M., von Buelow, P., and Stouffs, R. (2011), “Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms”, Advanced Engineering Informatics, doi:10.1016/j.aei.2011.07.009.

Turrin, M., von Buelow, P., Kilian, A., and Stouffs, R. (2011), “Performative skins for passive climatic comfort: A parametric design process”, Automation in Construction, doi:10.1016/j.autcon.2011.08.001.

Yuan, B. and Gallagher, M. (2007) “Combining Meta-EAs and Racing for Difficult EA Parameter Tuning Tasks”, in Lobo, F. J., Lima, C. F., and Michalewicz, Z. (Eds.), Parameter Setting in Evolutionary Algorithms, Series: Studies in Computational Intelligence, Vol. 54, 2007, XII, Springer-Verlag, Berlin, Heidelberg, pp. 121-142.

Zhang, T., Georgiopoulos, M., and Anagnostopoulos, G.C. (2013) “S-Race: a multi-objective racing algorithm”, in Proceedings of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, ACM, pp. 1565–1572.

Zitzler, E., Laumanns, M., and Bleuler, S. (2004) “A tutorial on evolutionary multiobjective optimization”, in Gandibleux, X., Sevaux, M., Sörensen, K., and T’kindt, V. (Eds.), Metaheuristics for Multiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems, Volume 535, 2004, pp. 3-37.