Achieving Optimal Illumination Conditions Using Local Search

Marios Sioutis
School of Information Science
Japan Advanced Institute
of Science and Technology
Nomi, Ishikawa, Japan
Email: smarios@jaist.ac.jp

Yuto Lim
School of Information Science
Japan Advanced Institute
of Science and Technology
Nomi, Ishikawa, Japan
Email: ylim@jaist.ac.jp

Yasuo Tan
School of Information Science
Japan Advanced Institute
of Science and Technology
Nomi, Ishikawa, Japan
Email: ytan@jaist.ac.jp

***Abstract*—In this research we attempt to achieve optimal illumination conditions in enclosed spaces that contain multiple illumination devices using local search algorithms. Given an illumination request, the search space which consists of the combination of all possible device settings is explored with the use of local search algorithms. The illumination effects of devices are modeled on an approach based on ray-tracing. Regarding the problem modeling, we consider multiple-objective evaluation functions from which a single fitness value can be extracted for a solution. Furthermore, we discuss the properties of the search operations that explore the search space and how they capture the need for both exploration and exploitation. The applied local search algorithms are evaluated in terms of convergence speed and solution quality.**

I. Introduction

In recent years, progress in consumer technology has reinvigorated the consumer field of home automation, by introducing more sophisticated, network-enabled home appliances at lower price points than ever before. Network communication protocols such as ECHONET Lite[1] and individual vendor platforms such as Insteon[2] and Apple HomeKit[3] allow the user to have control over the devices in the home. Nevertheless, such protocols and platforms only allow for some very rudimentary automation tasks to be carried out.

To realize the dream of a smart home, software services that incorporate intelligent decision making should be introduced in the home environment. Such smart services can automate tedious tasks on behalf of the user, resulting in a more comfortable everyday life. With the number of network-enabled devices constantly increasing, coordinating devices is going to become a daunting task for the user.

In this research, we focus specifically on a smart illumination service that can coordinate various illumination devices in order to achieve an optimal illumination setting requested by a user. Moreover, this service is able to handle conflicting requests from multiple users, attempting to fulfill each one of these request to the best degree possible. Such a service can also act as a subsystem of a home service platform that attempts to mediate conflicting operation of devices that influence the illumination in the house [4]. Conflicts on the environmental layer were discussed in [5], where illumination is such an aspect of the environment.

For this research, estimating the illumination effect a device has on the environment is critical. In our initial attempts, the popular Philips Hue [6] smart light bulbs are used as a base for modeling illumination devices.

II. Ray-tracing Based Device Modeling

Lux is a unit used to express the intensity of light as perceived by the human eye and it is equal to 1 lumen per square meter. As the distance from the illumination source increases, lux decreases according to the inverse square law. Using this relation, we proceeded to construct a ray-tracing based model for estimating lux in a give position in 3D space.

Each illumination device is modeled using a set of rays which subsequently bounce in a room. Collisions with the walls (and in future iterations, with furniture) are taken into consideration, with the energy of each ray reduced by a given absorption coefficient for each surface.

The 3D space is subdivided in smaller regions, with the use of an Octree. With each subdivision, a cube in depth d and size s is split into eight cubes at depth $d + 1$ and size $s' = s/2$. Given an initial octree node of size $s = 5m$, the resulting cubes at depth $d = 6$ end up being 15cm in size, a size good enough for our intended use. A ray-to-cube intersection is performed for each ray against the octree nodes in the scene, and the energy contribution of each ray is marked. As a final step, for each octree node, the energy contributions of rays are aggregated per illumination source. With the energy contributions per source now calculated and stored in memory for each cube, it is now possible to make estimations regarding the illumination at a given point in the 3D space of a room. A visualization of the ray-tracer can be seen in Fig. 1.

The relatively low number of rays used in the modeling process lead to some artifacts that are clearly visible. Adjacent cubes that are further away from the illumination sources may have significantly different illumination properties, due to the rays failing to clip one of the two cubes, as seen in Fig.2. To adjust for this phenomenon, a higher number of rays may be used. Furthermore, in the current model, all surfaces are treated as perfectly smooth (although they are associated with an absorption coefficient). This means that a ray will bounce cleanly from a surface, as if it was a mirror, with no scattering.

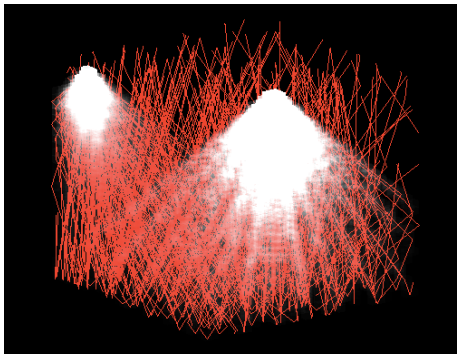

Fig. 1. Two lights (with rays)

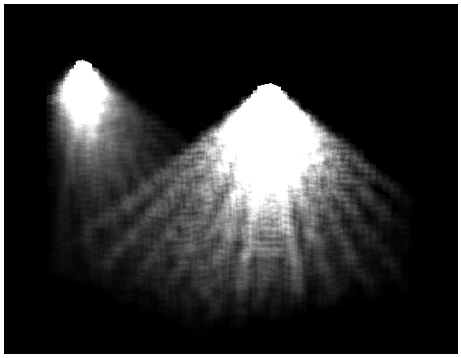

Fig. 2. Two lights (no rays)

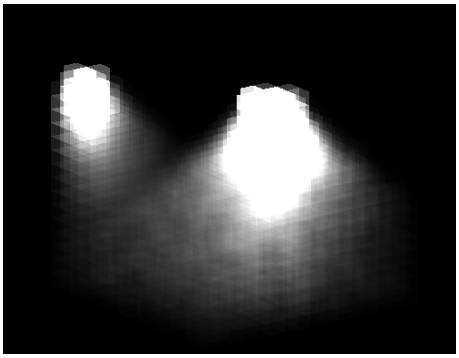

Fig. 3. Two lights (with smoothing)

To adjust for the above two facts, a "smoothing" pass can be performed, where adjacent cubes exchange a small amount of energy with each neighboring cube. The result is a smoother, more natural spread of the light in the scene, closer in resemblance to reality. This can be seen in Fig. 3.

III. Local Search Considerations

A. Candidate Solution Evaluation

The problem of achieving optimal illumination conditions in an indoor environment can be viewed as a continuous optimization problem [7]. A solution to this problem is evaluated in terms of the predicted illumination intensity at n points in physical space. Since illumination is a non-negative real number ($\mathbb{R}_0^+$), the solution space $\mathbb{P}$ is $\mathbb{R}_0^{+n}$. The estimated illumination at point i is denoted as p_i.

The search space $\mathbb{G}$ for the optimal illumination problem consists of all the possible brightness setting combinations of devices in a given space. Given a device i, the brightness setting for that device is denoted as g_i, with a range of $[0, 100]$.

In a scenario with m number of devices and n points in space for which illumination is to be estimated, a candidate solution takes the form $C = G, P$ where $G = \{g_1, g_2, ..., g_m\}$ and $P = \{p_1, p_2, ..., p_n\}$. Borrowing nomenclature from the genetic algorithms field, the prospective solution G of a candidate is usually called the *genome* and, respectively, P is called the *phenome*, (with g_i being a *gene* and p_i being a part of the phenome).

To evaluate a candidate solution, a *genome-to-phenome* mapping function of the form $P = gpm(G)$ is necessary. The proposed ray-tracer fulfills this task: given a set of device settings and a set of points in 3D space, it estimates the illumination at each given point.

Now that the phenome of a suggested solution can be determined, it must be evaluated for its suitability. The optimal illumination conditions can be specified as a set of objective functions $O = \{o_1, o_2, .., o_n\}$ where o_i is the objective function associated with point i. These objective functions can be used to denote lower or upper bounds for illumination at a given point. In the current implementation, these objective functions are subject to maximization. The objective function that specifies a minimum amount of illumination can be seen in Eq.1.

$$Over(val, target) = \begin{cases} -\frac{(val-target)^2}{target} & \text{if } val < 0, \\ \log_t(\frac{val}{target}) & \text{if } val >= 0. \end{cases} \quad (1)$$

By design, this function penalizes an illumination estimate val that is less than $target$. An estimation that exactly matches the target is going to yield a score of zero, and if the estimation is over the target a positive value will be reported. Furthermore, the relation

$$|Over(target - x, target)| > |Over(target + x)|$$

also holds true, i.e. positive results produce far less significant increase than their equivalent negative results. This feature allows the prioritization of unmet objectives over objectives that are met.

Finally, the results of the objective functions are aggregated through a fitness function, the result of which is used as the final evaluation of the candidate solution. Candidate solutions with higher reported fitness will prevail over candidates whose fitness is lower.

B. Search Strategies

With the infrastructure explained in the previous subsection in place, it is now possible to apply search strategies and explore the problem space.

Two metaheuristic search strategies where pursued: hill climbing and hill climbing with restarts, the details of which can be seen in Algorithm 1. In the case of pure hill climbing, the $shouldRestart()$ function always returns $false$. G represents the current candidate solution that will be used as a base for an exploration step. G' is the new genome that results after a mutation operation or a restart. Only solutions that improve over the best solution or the current solution will be remembered and considered for further exploration.

C. Mutation Operation

The mutation operation plays a very significant role in deciding the next candidate solution. Here, two different mutation operations where considered. The first mutation operation

Algorithm 1 Hill Climbing with Restarts

```
function HILLCLIMBING
    G, P, G', fit, fit', bestfit, bestG ← init()
    while terminate() == false do
        if shouldRestart() then
            G' ← Randomize(G)
            fit ← −∞
        else
            G' ← Mutate(G)
        end if
        P ← gpm(G'), fit' ← evaluate(P)
        if fit' > bestfit then
            BestG ← G', bestfit ← fit'
        end if
        if fit' > fit then
            G ← G', fit ← fit'
        end if
    end while
    return bestG
end function
```

is a completely random mutator that assigns a random valid setting to a random device present in the genome. Through pure luck, certain mutations will yield genomes that compare favourably to previous genomes, gradually improving towards a locally optimal solution. This mutator operation can be used as a control against more sophisticated mutator operations.

The second mutator operation examined is a binary search mutator. This optimized mutator operation picks a random gene and over the next few iterations tries to optimize it as much as possible. A new setting for that gene is decided and stored in local memory. If this setting survives until the next search cycle, the new gene produced a phenome with superior fitness. If the setting for that gene did not survive, it resulted in an inferior phenome. Using this information, the mutator can decide the direction of the binary search towards the most promising solution. When no further improvement can be made, the mutator's state is reset and in the next search iteration a new gene will be selected for optimization.

With two mutator operations and two metaheuristic strategies at hand, four possible combinations for an overall search strategy are possible: random mutator without restarts, optimized mutator without restarts, random mutator with restarts and optimized mutator with restarts. These four strategy combinations (hereby referenced as NR, NO, RR, RO) are the main focus of the experiment section that follows.

IV. Experiments and Results

A. Experimental Environment

The room used for the duration of the experiments can be seen in Fig. 4. The distribution of light sources in that room can be seen in Fig. 5. Starting from the left column and moving downwards, the lights are sequentially numbered starting from one. The second light of the first column is light number two, all the way to the last light being light number 25.

Fig. 4. Experimental Room

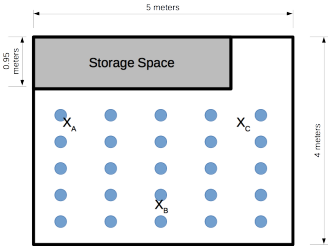

Fig. 5. Light Distribution

At points X_A and X_C the requested illumination is set to 300 lux, whereas at point X_B the requested illumination is set at only 20 lux. Using these request we applied the four search strategies explained earlier. For every experiment, the termination criteria were either the completion of 50000 search iterations or a total execution time of 1 second.

The restart strategy used was the lapse of 1000 search iterations without improvement. Finally, the experiments were conducted on a MacBook Air (2012 model), equipped with an Intel Core i7 2 Ghz processor, 8GB of memory and an SSD drive.

B. Algorithm Showcase

For each of the four search strategies proposed, an experimental session was conducted with the same random seed. The aggregated results can be seen in Tab. I. With the exception of the NO (no restarts, optimized mutator) search strategy, all other search strategies produced extremely similar results. More specifically, the algorithms correctly led to a solution in which the lights around the points X_A (lights 1, 2 and 6)and X_C (lights 16, 17, 21, 22 and 23) where necessary to fulfill the two out of three specified conditions, with all other lights having their brightness set to zero.

The similarity of the solutions is further emphasized by their fitness score which is the same up to 3 significant digits.

It must be stated that the estimated intensity reported by the gpm function that utilizes the ray tracer has a tendency to overestimate illumination. This can be attributed mainly to the accuracy of the ray tracer and to a lesser degree to the measuring protocol followed during the experiment.

C. Convergence Speed

To further understand the behaviour of the four algorithms, the experiment was repeated 200 times for each algorithm, using the same set of seeds.

The convergence speed of each algorithm was measured in iteration counts as well as time. These results can be seen in Fig. 6 and Fig. 7 respectively. From these results, it is readily apparent that the NO search strategy outperforms all other algorithms in terms of speed with a hefty margin. Average time for which the best solution was found was 79 milliseconds. The worst case scenario was 339 milliseconds.

In stark contrast, the NR (no restarts, random mutator) search strategy took on average 302 milliseconds with a worst

TABLE I
SHOWCASE RESULTS PER ALGORITHM

Search Strategy	Measured Illumination	Estimated Illumination	Fitness	Genome
NO	186, 14, 218	296, 20.1, 258	-0.01974	100 100 1 0 0 25 59 0 0 0 6 0 0 0 0 100 25 3 0 0 100 100 100 25 3
NR	194, 14, 226	291, 20.5, 279.5	-0.0062	100 100 0 0 0 69 0 0 0 0 0 0 0 0 0 100 77 0 0 0 100 100 100 0 0
RO	187, 14, 222	286, 20.7, 281	-0.00685	100 100 0 0 0 56 0 0 0 0 0 0 0 0 0 100 88 0 0 0 100 100 89 0 0
RR	195, 14, 227	291, 20.5, 279	-0.00622	100 100 0 0 0 69 0 0 0 0 0 0 0 0 0 100 76 0 0 0 100 100 100 0 0

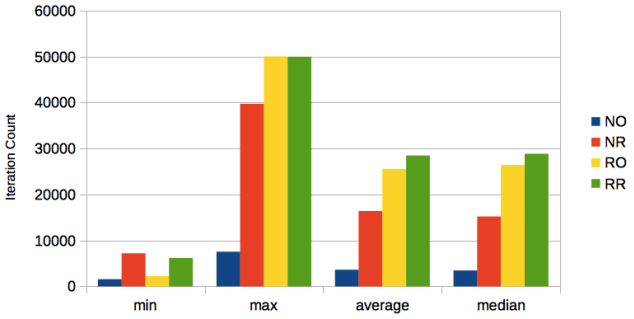

Fig. 6. Convergence (Iteration Count)

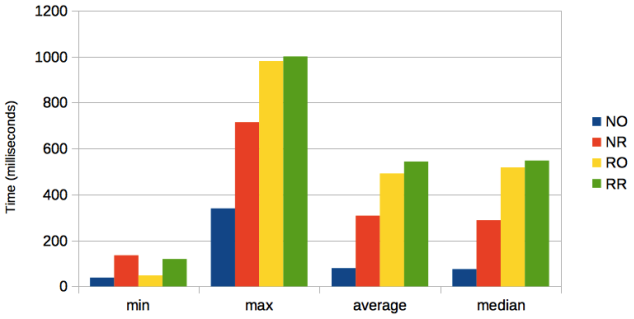

Fig. 7. Convergence (Time in milliseconds)

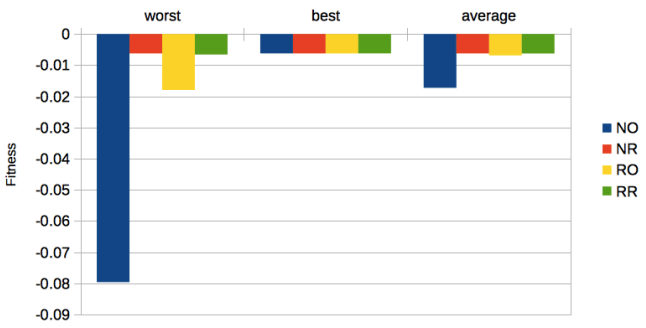

Fig. 8. Solution Quality

case scenario of 714 milliseconds. This is due to the non-probabilistic nature of the random mutator.

As expected, the average time for which the best solution was discovered for the RO and RR combinations was 490 milliseconds and 542 milliseconds respectively. The worst case results for both algorithms show that sometimes an optimal solution was found just before the time limit of one second was reached. The higher average time in which the best solution was reported is indicative of the restart strategy, but also demonstrates that this strategy actually led to a better solution after some restarts.

D. Solution Quality

The best, worst and average solution fitness for each algorithm can be seen in Fig. 8. It is of no surprise that the NO algorithm reports the worst solutions on average, trading solution quality for speed. The RO algorithm improves upon the NO algorithm with an average result that is close to the other two algorithms.

TABLE II
UNIQUE SOLUTIONS AND STANDARD DEVIATION

Algorithm	Unique Solutions	Standard Deviation
NO	162	0.012 011 9
NR	3	0.000 001 8
RO	71	0.001 236 503
RR	78	0.000 053 6

What may come as a surprise is that the average solution quality for the RR and NR algorithms is very close to the best solutions obtained. Specifically, as seen in Tab. II the standard deviation of the fitness for 200 runs is zero up to 4 and 5 significant digits respectively.

In absolute solution quality, the NR algorithm (no restarts, random mutator) comes out on top. In 200 runs the algorithm reported only 3 unique solutions that differed by one or two intensity points at two genes. This is a testament to the power of hill climbing and random mutations.

E. Comments

From the results presented in the previous sections, two points worth noting arise.

The first point is the accuracy of the ray tracer used as a gpm function. As explained in Sect. III-B, the ray tracer currently overestimates the illumination at given points. Although more accurate predictions would be preferable, its current behaviour is better than an alternative where underestimation would occur. The human vision has great adaptability, and variations in bright environments tend to go unnoticed by the user. In contrast, small variations in illumination in darker environ-

ments are far more pronounced and immediately perceived by the user.

The second point is about the proper selection of a search strategy. For our purposes, the case for selecting the NR algorithm can be made, based on the simple fact that it produced the most consistent results in terms of solution quality, with an exceptionally low standard deviation. Nevertheless, the average execution time for this algorithm is close to 500 milliseconds. Should a faster response time be of utmost importance, the use of the NO algorithm is advised.

V. Conclusions and Future Work

In this paper we discussed the use of local search algorithms for achieving optimal illumination conditions in enclosed spaces. To estimate the effects of illumination devices on the surrounding environment, a ray-tracer based approach was used.

Although the illumination estimations produced by the ray tracer leave a lot to be desired in terms of accuracy, by storing attenuation information in an octree enables the fast evaluation of the effects of illumination devices, enabling close to fifty thousand search iterations per second.

Regarding the use of local search algorithms, their convergence speed makes them an ideal fit for achieving optimal illumination conditions in indoor spaces. With convergence speeds of less than one second (including multiple restarts), local search algorithms can be used for real time control of illumination. Assuming that the location of a user moving inside a room can be tracked, a system utilizing local search algorithms can dynamically adjust the illumination of the room according to the user's position in real time.

As future work, in order to improve the illumination estimations, further adjustments that improve the accuracy of the ray tracer will be pursued, along with alternative methods for evaluating illumination. Regarding the local search algorithms, alternative objective and fitness functions that incorporate and combine heterogeneous information (such as the number of devices used, total energy consumption, etc.) in a single solution fitness will be pursued. The combination of such external information can drive the search towards solutions that prioritize certain characteristics over the estimated illumination intensity.

References

[1] ECHONET Lite Protocol Specifications. Accessed: 2015-06-12. [Online]. Available: http://www.echonet.gr.jp/english/spec/index.htm

[2] INSTEON. Accessed: 2015-05-28. [Online]. Available: http://http://www.insteon.com/

[3] Apple HomeKit. Accessed: 2015-05-28. [Online]. Available: https://developer.apple.com/homekit/

[4] M. Sioutis, J. Kim, A. Lim, and Y. Tan, "A home service deployment platform with support for detection and resolution of physical resource conflicts," in *Consumer Electronics (GCCE), 2012 IEEE 1st Global Conference on*, Oct 2012, pp. 333–336.

[5] M. Kolberg, E. Magill, and M. Wilson, "Compatibility issues between services supporting networked appliances," *Communications Magazine, IEEE*, vol. 41, no. 11, pp. 136–147, Nov 2003.

[6] Philips Hue. Accessed: 2015-06-12. [Online]. Available: http://www2.meethue.com/en-US/

[7] T. Weise, *Global Optimization Algorithms – Theory Application*, 2nd ed. Thomas Weise, 2008, Accessed: 2015-08-30. [Online]. Available: http://www.it-weise.de/projects/book.pdf