Optimization and Artificial Intelligence for Smart Devices

João Pedro Pedroso
INESC TEC and Faculdade de Ciências
Universidade do Porto, Portugal
E-mail: jpp@fc.up.pt

***Abstract*—The conquers of the last decades in the fields of artificial intelligence and optimization can provide tools for improving features in many applications for smart devices. We illustrate this by means of a case study where intelligent features, involving optimization problems, are considered for some devices. We explore challenges involved in their implementation, what kind of information is necessary for making the device truly intelligent, and possible ways of conveying it.**

I. Introduction

In many situations the decision that can be taken by a smart device can be described formally as mathematical optimization problem. In this context, if the data available is accurate, being *smart* is equivalent to being *optimal*, in the sense that if there are solutions better that those taken by the device, then it could have been smarter.

This paper lists some possibilities for improving the usability of devices by enabling them with optimization capabilities, using methods that are now a commonplace in scientific communities working either in optimization or in artificial intelligence.

II. Stand-alone devices

We start by considering a vacuum cleaner—an example which, though rather mundane, serves the purpose of illustrating cases where no external input is required, *i.e.*, the devices can operate in a stand-alone, independent fashion. Suppose that the device may clean a square at a time in Figure 1, and that we wish to minimize the distance travelled unnecessarily. For tackling this problem it is convenient to formalize it in mathematical notation; the first step is to prepare a graph describing the situation, as shown in Figure 2. One possibility for solving it is to assign a distance between each pair of nodes, and find a tour going through all nodes that minimizes the total distance traveled. This can be written as particular a case of the well-know, travelling salesman problem (under a degenerate, L_1 distance matrix; more realistic situations are instances of the *undirected rural postman problem*; see, *e.g.*, [1]). This problem is known to be NP-hard, meaning that there are no algorithms known to date that find the optimum in polynomial time, with respect to the input size; however, many heuristic and meta heuristic methods finding near optimal solutions have been proposed in the literature.

ISBN 978-4-89362-315-7 (C)2014 T-Engine Forum

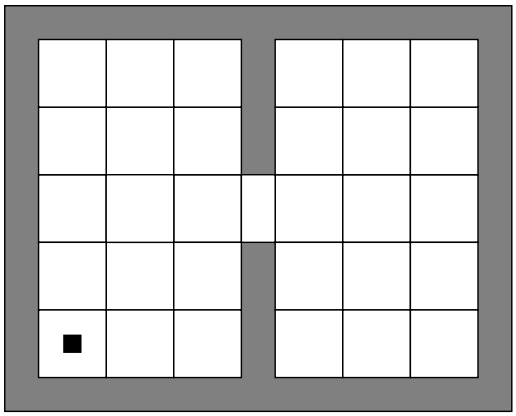

Fig. 1. Layout of a room for cleaning; a vacuum cleaner is represented as a black square, and is supposed to be able to clean one of the white squares at a time, at the same time that it advances.

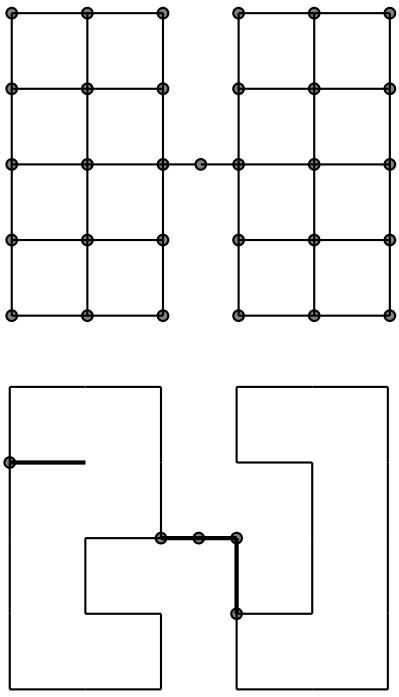

Fig. 2. Graph representation of the room (top). Nodes visited twice in an optimal solution (bottom).

The situation described in this section can be completely modeled and solved without resorting to the world outside the place where the device operates; hence, we classify this device as *standalone*. Its performance will depend solely on the capacity of capturing its our characteristics and those of a confined environment; and, of course, on a well-grounded method to solve the underlying optimization problem. What we would like to stress is that these methods—if not optimal, at least acceptable—are now commonplace in optimization science. Hence, the challenge is not on how to optimize but rather on how to integrate optimization in the devices, by providing the relevant information and collecting (and putting in practice) the solution produced.

As a conclusion, for **standalone devices:**

- all the relevant information can be captured without being connected to the outside world;
- performance depends only on their capabilities (including perception of the environment and optimization);
- in a more elaborated setting, user input or preferences may be used to fine tune the operation of the device.

III. Partly-connected devices

In our second example we introduce a conceivable device for optimally using the capacity of a battery (*e.g.*, of an electric car) when it is connected to a power supply. Let us describe a very simplified version this situation as a mathematical optimization problem. We discrete the span of the connection time in a set of periods $t = 1, \ldots, T$. Suppose that the power company provides energy at prices p_t and is ready to buy it from the battery at prices $v_t < p_t$; these are the external inputs that the device must acquire for optimal operation. The device is characterized by a charging rate r, a discharging rate d (if selling energy), a capacity Q, and the initial charge q_0; the charge available in the battery at period t is represented by q_t. The user defines a quantity $R \leq Q$ of energy that she requires to be available in the battery at the end of the planning horizon. The decision variables are, for $t = 1, \ldots, T$:

- x_t – binary variable with value 1 if the device is charging at period t, or 0 otherwise;
- y_t – binary variable with value 1 if the device is discharging (selling energy) at period t, or 0 otherwise.

We can now state the problem as follows:

$$\text{minimize } z = \sum_{t=1}^{T} (p_t r x_t - v_t d y_t) \tag{1}$$

$$\text{subject to: } q_{t-1} + r x_t = q_t + d y_t, \quad \text{for } t = 1 \ldots, T, \tag{2}$$

$$x_t + y_t \leq 1, \quad \text{for } t = 1 \ldots, T, \tag{3}$$

$$0 \leq q_t \leq Q, \quad \text{for } t = 1 \ldots, T, \tag{4}$$

$$q_T \geq R. \tag{5}$$

The objective is to minimize the net cost of charging the battery (1), subject to energy conservation (2), being either loading, unloading, or inactive (3), and capacity limits (4) in each period, while satisfying the restriction imposed on the final charge (5).

This model turns out to be easy to solve, *i.e.*, an optimal solution can be found in a very short time with a general-purpose mixed integer programming solver, for any reasonable input size. The difficulty is, once more, not in finding the optimum but in conveying the information needed for being able to do that.

A somehow similar situation, but considerably more difficult to model and optimize, concerns the minimization of the cost required for heating (or cooling) a house or building with time-dependent electricity prices. As in the case of charging a battery, one would like to use energy when it is less costly; but now the links between usage and result, in terms of the main output (temperature inside the building) are much more difficult to model. Furthermore, an important variable (outdoor temperature) cannot be forecast with certainty. Nevertheless, satisfactory models for solving this problem are available in the literature; see, *e.g.*, [2] for a model incorporating weather forecast information, and [3] for a model taking into account also room-to-room influences. A scheme of the information flow for using these methodologies is provided in Figure 3.

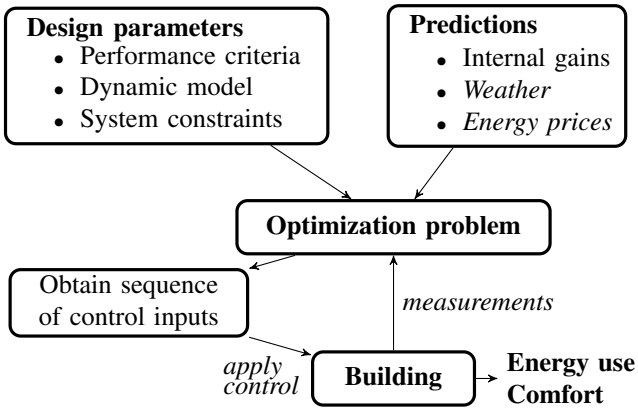

Fig. 3. Building climate control (adapted from [2]). Predictions for weather and energy prices must be dynamically acquired from external sources.

As a conclusion, for **partly-connected devices:**

- access to the outside world is required for capturing relevant information;
- performance depends both on the capabilities of the devices and on the accuracy of the data it obtained;
- providing and using user input for controlling the device is likely to be more difficult than for standalone devices: as this isn't a closed world, interactions are now more complex.

IV. Fully-connected devices

To illustrate the attributes of a fully-connected device we will use as an example an implement controlling traffic lights. Let us first focus on the situation of such an appliance

If there are several energy suppliers k from which the device can choose, it may compute the optimum z_k for each of them, and connect to the supplier k for which z_k is minimum (this is, hence, a min-min optimization problem).

operating in the countryside; in this case, we may consider that decisions taken by the device will only have a local impact (*i.e.*, the device does not affect traffic in other places). This being the case, if traffic is not congested it is trivial to make the device operate optimally: it should detect cars with sufficient anticipation, and whenever possible let them pass without avoidable stops.

When a set of traffic lights is connected as a network, each mutually influencing the others, their optimal operation becomes a much more difficult challenge. In simple settings, it would be desirable to have at least a mechanism for a device to detect if traffic is congested ahead, to use this information in its operation (reducing transit allowed into the congested area), and to send it to devices behind.

In more complex settings, it is questionable whether smart traffic lights would be enough to control the flow of vehicles. Most likely, a centralized control system is required; this allows making use of a model where the traffic policy is given as an input, and the system limits congestion in specified core areas of a city. Hence, with this example we have likely reached the current edge for the capabilities of a smart device. Still, the information that smart devices can provide to such a centralized control system is very relevant, and should be exploited; an interaction between central control and smart traffic lights is, presumably, the most promising direction for advancing usability in this field.

As a conclusion, for **fully-connected devices:**

- operation relies almost exclusively on information obtained from the outside world;
- performance clearly depends on the capabilities of the device, but it is also crucially dependent on trustworthy external data;
- most likely, many conflicting objectives have to be taken into account.

V. Artificial intelligence, optimization, and smart devices

In many situations, decisions that can be taken by a smart device are discrete: at a given step, either follow one path or follow another. This kind of decisions are well studied of combinatorial optimization, where the most common and successful methods rely on the exploration of a search tree. If the underlying problem is well defined, and the data unambiguous, then most likely a general-purpose solver for mixed integer programming can find an optimal solution (often very quickly [4]).

If the situation is less clear and a mathematical model cannot be readily derived, then most likely the appropriate method for finding a solution comes from the field of artificial intelligence. A recent method able to deal with such situations is Monte-Carlo tree search: an iterative procedure in which a search tree is asymmetrically constructed, attempting to expand its most promising parts, but balancing exploitation of known good branches with exploration of branches for which less information is available [5], [6]. The algorithm, used very successfully in game playing, is based on the idea of Monte-Carlo evaluation: the reward associated with a particular node is estimated from the results of random simulations started from that node. Each node keeps track of the number of simulations run from its state as well as their outcomes, and these data are used to produce an estimate of the value for the node when deciding how to expand the tree. This can be trivially extended to the case where real world information is used to complement outcomes generated with simulation (with much more relevance given to real data).

Other situations deal with decisions on continuous values (*e.g.*, the time allowed to a certain path in a given traffic light), typically in the context of nonlinear optimization. Concerning smart devices, it expected that most of the relevant cases can be handled by derivative free methods; the requirement for the optimization process is to provide a method capable of quickly finding optima, or very good solutions, without information on derivatives (see, *e.g.*, [7]). Interesting properties concern being able to swiftly reoptimize upon changes on data, and to deal with noise; several methods for this are proposed in [8]. One of the favorites in the engineering community, due to the simplicity of implementation, is the simplex method for nonlinear optimization [9].

A very important requirement for providing optimization technology in smart devices is to have access to a simulated environment, where an expected outcome is generated for a given input (*i.e.*, for a tentative solution of the optimization method) without actually putting it in practice. This simulation step is essential for training the optimization method, so that its solution can move towards interesting areas without going through costly (or otherwise undesirable) decisions in the real world. Simulation tools are also a requirement for prior training in factory, before the device is put in its operating environment; the aim is again to avoid learning with costly decisions in the real world.

As a summary, the challenges to the device design community are the following:

- to produce devices aware of the capabilities of the recent progress in artificial intelligence and optimization;
- to provide a "simulation mode", representing the outside world without actually interacting with it;
- to prepare mechanisms for prior training in factory, for having a reasonable solution when the device starts operating in practice;
- a lot of work must be done on reality perception, data detection, and data cleanup.

As for the optimization community, the challenges can be summarized as follows:

- to provide appropriate heuristics and best practices for the cases where the optimization problem is known to be difficult;
- to fulfill the requirement of robust and flexible algorithms, capable of adapting to many different situations;

Taking into account the social savings that could be attained, this is a much neglected improvement opportunity.

- fool proof design: guarantee that unacceptable solutions will not be proposed when the device is brought into operation in the real world.

VI. Conclusions

Optimization methods and software have evolved into a mature subject, providing algorithms and tools that can be used to solve a number of real life problems, or to improve the operation of commonly used devices. However, these methods are still mostly confined to applications in organizations, being largely absent from everyday life — even considering state-of-the-art smart devices.

In this short paper we provide some examples of applications of optimization methods in typical, everyday routine, and of the challenges involved. The paper deals mainly with situations that, to the best of our knowledge, are not being addressed with the depth that they could, and hence the current outcomes are likely to be suboptimal. The paper has raised several questions and provided no answers. Our expectations are that the issues raised will be addressed by joint efforts of researchers coming from two fields: device design and optimization; nevertheless, the interdisciplinary nature of these subjects may require the support of other areas.

References

[1] H. A. Eiselt, M. Gendreau, and G. Laporte, "Arc routing problems, part II: The rural postman problem," *Operations Research*, vol. 43, no. 3, pp. 399–414, 1995.

[2] F. Oldewurtel, A. Parisio, C. N. Jones, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and M. Morari, "Use of model predictive control and weather forecasts for energy efficient building climate control," *Energy and Buildings*, vol. 45, no. 0, pp. 15 – 27, 2012.

[3] C. Ellis, M. Hazas, and J. Scott, "Matchstick: A room-to-room thermal model for predicting indoor temperature from wireless sensor data," in *Proceedings of IPSN 2013*. ACM, April 2013.

[4] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. D. Mittelmann, T. K. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter, "MIPLIB 2010," *Math. Program. Comput.*, vol. 3, no. 2, pp. 103–163, 2011.

[5] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in *Proceedings of the 5th International Conference on Computers and Games*, ser. CG'06. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 72–83.

[6] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[7] L. Rios and N. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013.

[8] A. Gosavi, *Simulation-based optimization: parametric optimization techniques and reinforcement learning*, ser. Operations Research/Computer Science Interfaces Series. Springer, 2003, vol. 25.

[9] M. H. Wright, "Nelder, Mead, and the other simplex method," *Documenta Mathematica*, pp. 271–276, 2012, special volume ("Optimization Stories").