Client-side combinatorial optimization

Vittorio Maniezzo[1,a] , Marco A. Boschetti[2], Antonella Carbonaro[3],
Moreno Marzolla[4], Francesco Strappaveccia[5]

[1] University of Bologna, via Sacchi 3, 47521, Cesena, Italia
vittorio.maniezzo@unibo.it

[2] University of Bologna, via Sacchi 3, 47521, Cesena, Italia
marco.boschetti@unibo.it

[3] University of Bologna, via Sacchi 3, 47521, Cesena, Italia
antonella.carbonaro@unibo.it

[4] University of Bologna, via Sacchi 3, 47521, Cesena, Italia
moreno.marzolla@unibo.it

[5] University of Bologna, via Sacchi 3, 47521, Cesena, Italia
francesco.strappaveccia@unibo.it

1 Extended Abstract

Real world applications of advanced optimization algorithms have traditionally been run on dedicated desktop machines. More recently, client server architectures have become a common viable alternative to standalone installations, providing access both via web apps and via web services (see for example [1,2] among the many possible references). Obvious benefits of this choice come from the possibility to provide access to a wider set of users, or from a wider range of locations, and to invest on a single high-end server machine that ensures good performance reducing maintenance costs, just to name a few.

While this distributed client server architecture is satisfying in many actual use cases, there are situations where its deployment requires more resources than those that can be made available. This can be true with reference to the computational power of the server machine, though this problem can usually be easily abated, but more importantly with reference to the network connection. In fact, in all cases where the basic IP connection is unreliable, the client cannot commit to the server the required computations. Actual industrial cases where this happens are rare, but in some settings unavoidable. Some examples are:

1) Loading. Factory production environments are characterized by intense magnetic fields, which deny the possibility of using a wi-fi connection. If trucks are to be loaded inside or in the proximity of a production plant, and wired connections are not an option, it becomes difficult to provide the layout planner with a load optimization code to assist him dictating loading directives.
2) Packing. A use case arises when a commercial agent visits a prospect customer, proposing bulky and possibly irregularly shaped items. There could be a common interest to fill up one, or an integer number of, container or truck with the customer order, so to minimize the impact of fixed costs. In order to ascertain this, it is necessary to solve a 3D nesting problem, with no guaranteed connection at the customer's premises.
3) Warehousing. Large warehouses are usually uncovered by wi-fi, and there is evidence that an optimized online rearrangement strategy can significantly reduce picking costs, which in turns represent the largest component of the internal logistic costs [3]. It is beneficial to provide forklift operators with the support of an optimizer for the *online block relocation problem* that he is implicitly called to solve each time an unforeseen movement order arrives.
4) Routing. Maintenance companies usually have a number of agents or teams of agents working in the territory. Each one is assigned a set of customers and thus must implement a solution to a

[a] Corresponding author.

TSP with time windows (TSPTW) on them. Often new customers are assigned during the shift, or additional visits are tentatively inserted by the agents themselves, in situations where the data connection permits to reliably receive a message stating of the new assignment, but not to be kept up for all the duration of the re-optimization, especially taking into account the manual editing of the solution which is often requested by the agents upon checking the solution.

1.1 Platform maturity

Despite these possibilities, very little effort has been put in delivering optimization code into front end components. This was surely justified by the lack of computing power of mobile platform until very recent times, but it is ever more questionable with the rapid increase of computational power of mobile phones and tablets, as reported in table 1. It is well known, for example, that a single Apple iPhone 5 has 2.7 times the processing power than the 1985 Cray-2 supercomputer.

Year	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
tot freq	0.4 (0.4)	0.2 (0.2)	0.4 (0.4)	0.4 (0.5)	0.5 (0.8)	0.6 (1.0)	0.8 (1.0)	1.5 (3.0)	3.6 (6.8)	5.4 (9.2)	7.7(12.8)	11.5(14.4)	8.1(10.8)
core freq	0.4 (0.4)	0.2 (0.2)	0.4 (0.4)	0.4 (0.5)	0.5 (0.8)	0.6 (1.0)	0.8 (1.0)	1.1 (1.5)	1.4 (1.7)	1.7 (2.3)	1.9 (2.5)	1.7 (1.9)	1.9 (2.5)
num cores	1.0 (1)	1.0 (1)	1.0 (1)	1.0 (1)	1.0 (1)	1.0 (1)	1.0 (1)	1.4 (2)	2.5 (4)	3.0 (4)	4.1 (8)	6.8 (8)	4.4 (8)
RAM	0.1 (0.1)	0.1 (0.1)	0.1 (0.1)	0.1 (0.1)	0.2 (0.4)	0.2 (0.3)	0.5 (0.8)	0.7 (1.0)	1.2 (2.0)	1.9 (3.0)	2.0 (3.0)	2.9 (4.0)	3.4 (6.0)

Table 1: Increase in mobile cpu's computational power, average (max) values, from [11].

Furthermore, the increased ripeness of mobile platforms as target for the deployment of optimization codes derives also from the increased possibilities offered by software middlewares. The last decade has been in fact characterized by the approach of delivering solutions via apps. This is extremely functional in many settings, but it is restrictive for industrial optimization as it imposes to develop and maintain different codes for the different targeted platforms. The advantage of deployment as apps derived mainly from the necessity of reverting to native applications in order to make full use of the platform hardware components. However, the recent enhancements of mobile browser possibilities and the availability of advanced multiplatform development technologies, made browsers a plausible alternative to native apps [5,6]. This is especially true for optimization applications, where there is limited need for many specialized hardware components, such as camera, ambience light detection or Bluetooth transfers. It is now in fact possible to make full use of the platform computational power, including multithreading on multicore CPUs, and access to wide memory areas and to local storage. It is therefore possible to develop and deploy one single solution, which can be accessed on desktops, tablets and phones, be they iOS or Android, and used anywhere. This last feature is granted by different mobile technologies, such as HTML5 application cache or Service Workers [7]. The solution will be deployed and maintained on one single server, as in traditional back end applications, but run locally on the client machines.

1.2 Algorithmic challenges

The possibility to distribute the optimization module along with the front end gives rise not only to a market opportunity, but – more interestingly for our research community – also to algorithm design challenges. Industrial applications have been mainly attacked with heuristic or metaheuristic approaches, more recently with matheuristics [9,10]. The effectiveness of these approaches has been certified primarily by the dedicated research community, which designed and tested them in laboratories on suitably powerful machines. Despite the dramatic increase of computational power shown in Table 1, mobile platform are still significantly inferior to a computer science lab workstation, therefore a decrease of the effectiveness expected on a dedicated desktop is in order.

Moreover, optimization middleware is still largely unavailable for distributed offline frontends. This means that the optimization code must be all self-contained and, for the time being, essentially fully developed for the application. This has little impact on standard metaheuristics, such as tabu search,

VNS or GRASP, which are usually designed like that anyway, but it has much impact on matheuristics.

Matheuristic approaches have been boosted by the increasing effectiveness of MILP solvers, and many new proposals include a LP or a MIP as a subproblem to solve. These approaches are to be ruled out, as at the moment there are no effective LP-solver available as javascript libraries (none of the current ones can scale above a few thousands decision variables), let alone MIP solvers.

This leaves however room for quite a variety of approaches [9,10], including Lagrangian heuristics, beam search, forward and backward, corridor methods, dynasearch among others.

We started a computational testing, comparing a number of metaheuristic and matheuristic approaches on generalized assignment problem instances taken from [8]. All codes were run on different platforms. Table 2 presents preliminary results, paired with those obtained by the very same algorithm implemented in C++ and run on the same 3.50 GHz, Intel I7-4771 machine also used in column 4. The table reports the time, in ms, of a Lagrangian heuristic, where termination condition was set on the step size coefficient in the inner subgradient, stopping when it got less than 0.01. Results obtained with other methods will be presented at the conference.

instance	n	m	Browser, I7	Windows ph. 8.1	Huaway Android 4.2	iPhone 5 iOS 10.2	C++
elba	40	5	103	512	823	842	19
gap8_0	48	8	126	1391	1205	1089	38
gap12_0	60	10	189	2396	2330	2533	128
gapa_0	100	5	35	224	316	416	13
gapd_0	100	5	217	2621	2617	4383	140
e20200	200	20	6440	121865	86814	113674	2990

Table 2: Generalized Assignment Problem validation (wall clock times in ms).

References

[1] NEOS server, https://neos-server.org/neos/, accessed January 2017.

[2] EPS2016: EURO PhD School 2016, https://sites.google.com/site/eps2016matheuristics/, accessed January, 2017.

[3] Tompkins, J.; White, Y.; Bozer, E.; and Tanchoco, J. 2010. Facilities planning. John Wiley & Sons.

[4] Processing power compared. http://pages.experts-exchange.com/processing-power-compared/ accessed January 2017

[5] Stéphanie Walter, The (Not So) Secret Powers Of The Mobile Browser, Smashing Magazine, https://www.smashingmagazine.com/2016/12/the-not-so-secret-powers-of-the-mobile-browser/

[6] Google inc., Progressive Web Apps: A new way to deliver amazing user experiences on the web. https://developers.google.com/web/progressive-web-apps/, accessed January 2017.

[7] Google inc., Service Workers: an Introduction, https://developers.google.com/web/fundamentals/getting-started/primers/service-workers, accessed January 2017.

[8] Bridging the GAP, Some Generalized Assignment Problem instances http://astarte.csr.unibo.it/gapdata/gapinstances.html, accessed January 2017.

[9] Maniezzo, V., Stützle, T., Voß, S. (Eds.), Matheuristics, Hybridizing Metaheuristics and Mathematical Programming, Annals of Information Systems, Springer, 2010

[10] Boschetti M.A., Maniezzo V., Roffilli M., Bolufé Röhler A. (2009) Matheuristics: Optimization, Simulation and Control. In: Blesa M.J. et al. (eds) HM 2009, Lecture Notes in Computer Science, vol 5818. Springer

[11] Comparison of smartphones, https://en.wikipedia.org/wiki/Comparison_of_smartphones, accessed January 2017.