Exploring Continuous Optimization Solutions for Business-driven IT Management Problems

Mauro Tortonesi
Department of Engineering
University of Ferrara
Ferrara, Italy
mauro.tortonesi@unife.it

***Abstract*—Business-driven IT management practices often involve the performance optimization of a system according to business criteria. The increased attention to dynamical aspects of the system behavior, the preference for simulative approaches rather than analytical ones, and the increased level of complexity posed by business-driven performance evaluation significantly complicate the optimization of BDIM systems and demand a radical rethinking of methodologies and tools. This raises the opportunity to devise and implement common methodology and tools that could be used for a large class of different BDIM optimization problems. This paper proposes a generic framework for the dynamic and adaptive optimization of BDIM systems, introduces the Open Source *ruby-mhl* metaheuristics library, and provides an experimental evaluation in the context of a realistic case study.**

***Keywords*—Business-driven IT management (BDIM), IT service management, optimization**

I. Introduction

Business-driven IT management (BDIM) is a practice in IT service management that attempts to evaluate and optimize the IT infrastructure according to business criteria [1]. BDIM has the ultimate objective of studying and optimizing an IT service from the point of view of the service provider's business management, measuring both tangible and intangible business costs.

Despite their advantages, BDIM approaches present some peculiar challenges in optimizing a system operating conditions. In fact, BDIM researchers need to deal with optimization problems that involve complex solution spaces and target functions that are rather expensive to compute (usually involving one or more simulation runs), thus making traditional gradient-based optimization solutions unfit. In addition, BDIM systems have non-trivial dynamical aspects and their behavior changes over time.

There is the need to explore optimization solutions that are better suited than traditional mathematical optimization to BDIM problems. More specifically, BDIM problems call for continuous and adaptive optimization solutions, capable of effectively exploring the large and complex space of feasible system configurations to find the most convenient ones, and of re-tuning the trade-off between exploration and exploitation.

For the purpose of this paper, *we define as BDIM optimization problems the subset of problems in the IT service management research area that aim at optimizing the configuration, i.e., the set of operating parameters, of a system according to business criteria*. The similarities between these problems suggest the opportunity to investigate common methodology and tools that could be applied to the widest range of different BDIM systems. The present paper aims to be a first step in this direction.

II. A Framework for Continuous Optimization in BDIM

We introduce a framework designed to consider all the fundamental characteristics that BDIM optimization problems exhibit: dynamic inputs; different (time-varying) operating conditions; complex business impact evaluation procedures; and reference configurations. The framework, whose architecture is depicted in Fig. 1, implements an automated continuous optimization of system, involving a feedback between the optimization procedure and the system.

The Demand Model component generates a demand trace according to user-specified rules, adopting stochastic processes for the reenactement of the demand arrival process [2]. The System Model component takes care of reenacting the system and measuring how it would behave under different working conditions. The Business Impact Analysis component is in charge of evaluating the business impact of the current system configuration x. The Optimization component implements the optimization procedure, implementing algorithms that export a set of parameters θ that control the behavior of the optimization procedure. By changing the value of θ, the optimization algorithms can be tuned towards a more exploitative or a more explorative behavior.

The Optimization component embeds a Controller module that modulates θ according to the performance (recently) exhibited by the optimization procedure, with a feedback control loop. For instance, in case the optimization algorithm reaches a "plateau" in the possible solution space, in which no improvement to the current solution could be found in several past iterations of the algorithm, the Controller module could decide to change the parameters θ to alter the behavior of the optimization algorithm to favor the exploration of different portions of the solution space. Instead, if the controller detects a significant improvement in the last iterations of the algorithm related to a the exploitation of a specific portion of the solution space, it could decide to change the parameters θ to alter the behavior of the optimization algorithm to favor the local search close to the best solutions recently found ("exploitation").

As a new solution x is found, it is fed to a Decision

Making component that might decide to change the operating conditions of the system to improve its performance. The Decision Making component would be ideally connected to an actuator that could change the current system from x_0 to x (also resetting the value of x_0 parameter).

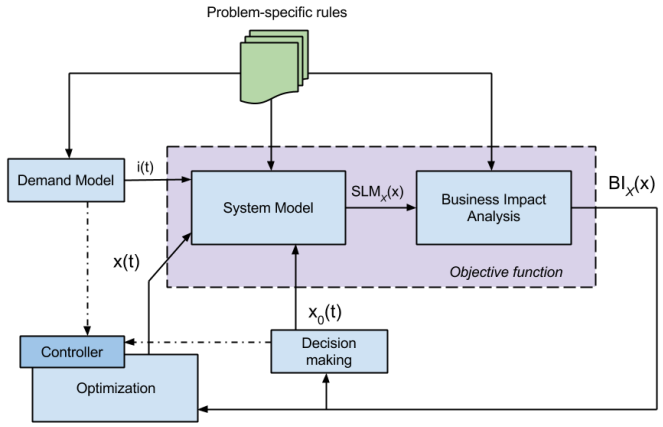

Fig. 1. A framework for continuous optimization in BDIM

III. Metaheuristics for BDIM Optimization

Genetic Algorithms (GAs) [3] and Particle Swarm Optimization (PSO) [4] based metaheuristics are particularly well suited for the optimization of dynamic and non-smooth target functions.

Genetic Algorithms are inspired to biological evolution: they define a population of individuals, each one representing a specific coordinate on the search space and having a "fitness" value. Through mating operators, including selection, recombination, and mutation, the genetic material is evolved through several generations, to find the fittest individuals that represent better solutions to the problem at hand. GAs have several desirable properties in BDIM optimization. By acting on the parameters that control the behavior of their mutation and selection operators, GAs can relatively easily be tuned towards more explorative or more exploitative behaviors [5].

PSO is a swarm intelligence technique inspired to the behaviour of bird flocks, which also integrates rather well within a continuous optimization framework [6]. Traditional PSO is a relatively simple to implement optimization algorithm that, however, unlike GAs presents a few critical aspects, such as lower resilience to early convergence [4] and a more difficult parameter tuning process. Improved versions of the algorithm such as Quantum-based PSO and variants of PSO based on multiple swarms have later emerged to addressed these issues.

Population-based metaheuristics, such as GAs and PSO, are well suited for the optimization of dynamic systems, like BDIM ones. In fact, the population maintained by the algorithms represents the "memory" of the search process. In case of changes in the system input or inner working (such as when one or more support groups in an IT support organization transition from daily to nightly shift), the population members of the last generation evaluated represent a good "priming" for the next optimization round. At the same time, the exploration capability of GAs and PSO can be ensured by the adoption of operators that increase the population diversity.

TABLE I. Transition matrix used in the experiments

From/To	SG1	SG2	SG3	SG4	SG5	SG6	SG7	SG8	Out
In	1	2	43	2	2	3	2	1	0
SG1	0	2	2	6	1	13	1	4	2
SG2	2	0	3	4	2	3	6	1	1
SG3	5	21	0	2	3	34	2	2	4
SG4	1	4	4	0	2	3	2	1	7
SG5	2	2	1	5	0	1	1	3	3
SG6	3	2	6	0	2	0	2	4	1
SG7	1	8	3	2	2	3	0	1	2
SG8	3	8	1	0	5	1	34	0	14

TABLE II. Support group configuration for the experiments

Name	Timezone	Mean service time (seconds)
SG1	BRST	1114.00
SG2	BRST	1980.00
SG3	EDT	4099.00
SG4	CEST	4979.00
SG5	CEST	2760.00
SG6	IST	7631.00
SG7	IST	2170.00
SG8	IST	9220.00

IV. Experimental evaluation

To evaluate how our framework, we created a realistic IT support organization model and we set up to its optimization. We considered an enterprise class service, with 8 support groups dislocated in 4 locations: Brazil (2 support groups), east coast of the USA (1 support group), central Europe (2 support groups), and India (3 support groups). We assumed that in every support group operators worked a 8 hour shift, from 9AM to 5PM local time.

We adopted *Symian*, a state-of-the-art simulator which we developed in the context of our research [7] and that we recently released as Open Source at https://github.com/mtortonesi/symian. *Symian* models IT support ogranizations as an open queuing network, renacting each support group as a queue with a specific service discipline and the incident escalation process through a Markov transition probability matrix. More specifically, for these experiments we used the (unnormalized) transition probability matrix presented in Table I. In the attempt of producing an experiment as realistic as possible, the matrix was randomly generated by sampling from a reshaped empirical probability distribution obtained from a dataset containing information on incidents from a real-life enterprise class IT support organization.

Support groups are modeled as a G/M/s-FIFO queue with a specific mean service time [7]. In turn, operators have different annual salaries: $\$\,30,000.00$ for the operators working in Europe, $\$\,25,000.00$ for operators working in USA, $\$\,20,000.00$ for operators working in Brazil, and $\$\,15,000.00$ for operators working in India. We do not consider other costs, such as equipment for operators, that we assume are significantly smaller and thus could be ignored for the purpose of these experiments. Table II summarizes the support groups characteristics. For contracting costs, we considered the SLA defined in Table III.

We also considered an additional component to evaluate the cost of performance drifts in system configuration:

TABLE III. SLA CONSIDERED FOR THE EXPERIMENTS

KPI	Target	Penalty (per month)
Mean Time To Resolution (TTR)	$\leq 10days$	$\$\,300,000.00$
Max Time To Resolution (TTR)	$\leq 28days$	$\$\,360,000.00$

$$C_{drift} = \$\,120,000.00 * \frac{2}{\pi} \arctan\left(\frac{m_{micd} - target_{micd}}{10 * target_{micd}}\right)$$

where m_{micd} is the value of the number of incidents closed per day metric obtained when evaluating the new IT support organization configuration and $target_{micd}$ is the reference value of the same metric, obtained from the evaluation of the configuration currently in place. By evaluating the system in the reference configuration with 10 simulation rounds we obtained a value of $target_{micd} = 31.5$, which we used for all the experiments.

To reenact the incident arrival process, we adopted a stochastic model based on the Generalized Pareto distribution:

$$F_{(\xi,\mu,\sigma)}(x) = \begin{cases} 1 - \left[1 + \frac{\xi(x-\sigma)}{\sigma}\right]^{-\frac{1}{\xi}} & \text{for } \xi \neq 0 \\ 1 - e^{-\frac{x-\mu}{\sigma}} & \text{for } \xi = 0 \end{cases}$$

and we selected the location $\mu = 1000.0$ (seconds), scale $\sigma = 200.0$ (seconds), and shape $\xi = 0.75$ parameters in order to have a mean time between two incident arrivals of 1800 seconds (30 minutes), and infinite variance. We chose this model because it represents an acceptable tradeoff between simplicity and realistic modeling [2].

A. Results

We first run a baseline experiment to optimize the IT support organization using GAs and PSO. To this end, we adopted a rather straightforward integer vector representation for the population individuals used in the algorithms. More specifically, each individual was represented as an integer vector whose cells contain the number of operators allocated for each support group. For GAs, we used populations of 100 individuals and stopped the optimization process after 20 generations. For PSO, we considered a swarm of 40 particles and stopped the optimization process after 50 generations. The GA and PSO implementations used in these experiments are available in the *ruby-mhl* library, that we released as Open Source: https://github.com/mtortonesi/ruby-mhl.

We used two different versions of GAs in the optimization, in both cases adopting a mutation operator that randomly samples from a geometric distribution. In the first version we used a fixed mutation probability parameter $p_m = 0.4$ while in the second version we used an adaptive mutation probability parameter, controlled by the Rechenberg algorithm [6].

Fig. 2 and 3 present the results we obtained with GAs and PSO respectively. The figures depict the business impact calculated on the best sample of the population corrisponding to the generation indicated in the x axis. Fig. 2 shows that, despite the luckier start of the GA version with fixed mutation probability (we primed the two GA optimization runs using different randomly sampled populations), the GA version using adaptive mutation probability seems to exhibit a more consistently improving behavior. However, Fig. 3 shows that PSO performs much better than GAs in this particular optimization problem, as it converges towards a better optimal solution (with a value of $\$22,493$ compared to $\$28,712$ and $\$24,643$ returned by GAs) and with a significantly faster pace.

We then launched a second experiment, simulating a 5% increase in demand with respect to the baseline conditions of the first experiment, by changing the location parameter of the GPD distribution to the value $\mu^* = 914.29$, while preserving the previous values for the scale and shape parameters. This time we used three different versions of GAs. In the first version, we set fixed mutation probability $p_m = 0.4$ while in the second version we adopted a hypermutation strategy to quickly increase population diversity in the attempt to speed up the optimization process. In both cases, we primed the algorithm using as the initial populations the ones evolved as the last generation in the two baseline optimization experiment runs, with fixed and adaptive mutation probability respectively.

Hypermutation is a dynamic optimization strategy developed in the context of immune artificial systems research and inspired to the behavior of biological cells, that tend to aggressively increase their mutations in response to stressful changes in their environment [8]. The hypermutation strategy we implemented in our GA responds to changes by immediately setting the value of the parameter to a more aggressive value of the mutation probability parameter (in our case $HM = 0.3$), thus increasing the intensity of mutations and causing the GA population to enter a "hypermutation" state. As reported in literature [9], this is a relatively simple but very effective strategy. After resetting the mutation probability parameter to the HM threshold, the GA resumed running with an adaptive mutation probability parameter modulated by a Rechenberg controller.

Finally, in the third version of GA we adopted a naive approach restarting the algorithm from a randomly sampled population and using a fixed mutation probability parameter $p_m = 0.4$.

We also used two different versions of PSO. In the first one we primed the algorithm using as the starting particle swarm the one obtained from the last generation of the PSO algorithm in the baseline optimization experiment, and priming the algorithm from a random particle swarm. In both cases, we inizialized the particles with random velocities.

Fig. 4 and 5 present the results we obtained with GAs and PSO respectively. Both the figures clearly demonstrate the advantage of priming the algorithms with a population obtained from the optimization of the system when operating in a previous state. Fig. 4 also shows that the adoption of the hypermutation strategy makes the GA relatively quicker in finding better locations in the search space at first. However, let us note that the GA version with random population restart exhibited a slower improvement pace but higher search resolution, ending up exploring a portion of the search space that proved better than the ones explored by the primed versions of the GA.

Finally, PSO significantly outperformed GAs in this experiment as well, obtaining a best configuration corresponding to

a business impact of $\$5,863$ against the $\$21,041$ of the best configuration returned by GA. This motivates us to further investigate the adoption of PSO in our future research, also exploring adaptive and advanced versions of the algorithm.

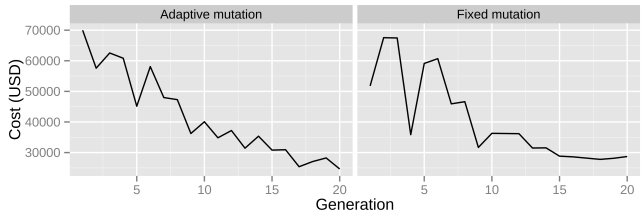

Fig. 2. Results of the baseline optimization using GA.

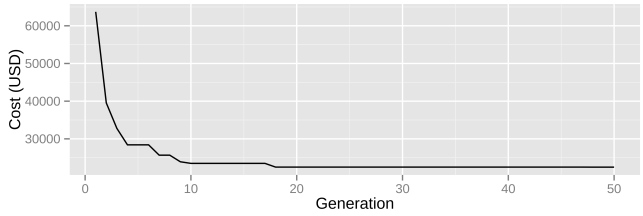

Fig. 3. Results of the baseline optimization using PSO.

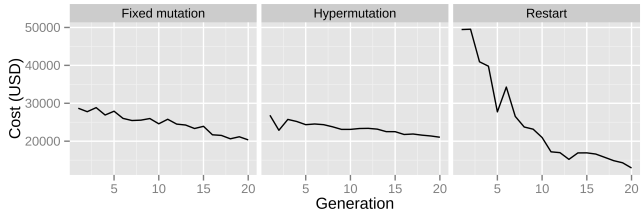

Fig. 4. Results of the optimization with 5% higher workload using GA.

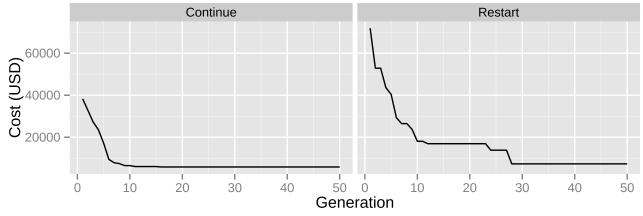

Fig. 5. Results of the optimization with 5% higher workload using PSO.

V. Related Work

In network and system management, researchers are increasingly turning towards computational intelligence methods for the optimization of the systems they study. The formerly cited works [10] and [11] attempt to optimize the placement of software components in federated Cloud environments, respectively using a traditional GA and a memetic algorithm based on GA with integer vector representation.

A recent work by Moens et al. presents an interesting comparison between a traditional ILP optimization method with genetic algorithm and particle swarm optimization [12]. The authors conclude that the guarantee of finding the best solution of the problem provided by ILP is well offseted by its scaling issues as the problem dimension grows larger.

However, most of those works focus on the (static) optimization of systems operating in steady-state, and do not consider the dynamic system behaviors. This paper represents a modest attempt to push forward the state of the art, by introducing a framework and releasing software tools for dynamic optimization problems, designed for BDIM but broadly applicable to other areas of network and system management.

VI. Conclusions and Future Works

Optimization solutions based on computational intelligence methods represent very interesting tools for BDIM, and for many other optimization problems in network and service management research. There is the need to investigate further the possibilities offered by the adoption of those methods in a framework for dynamic and adaptive optimization such as the one presented in this paper. In particular, robust and advanced control solutions to tune the parameters of optimization algorithms are a very interesting research avenue.

References

[1] A. Moura, J. Sauve, and C. Bartolini, "Business-driven IT management - upping the ante of IT: exploring the linkage between IT and business to improve both IT and business results," *Communications Magazine, IEEE*, vol. 46, no. 10, pp. 148–153, October 2008.

[2] C. Bartolini, C. Stefanelli, and M. Tortonesi, "Synthetic incident generation in the reenactment of IT support organization behavior," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 126–133.

[3] S. Luke, *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013, available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[4] J. Sun, C.-H. Lai, and X.-J. Wu, *Particle Swarm Optimisation: Classical and Quantum Perspectives*. CRC Press, 2011.

[5] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 35:1–35:33, Jul. 2013.

[6] O. Kramer, *A Brief Introduction to Continuous Evolutionary Optimization*. Springer, 2013.

[7] C. Bartolini, C. Stefanelli, and M. Tortonesi, "SYMIAN: Analysis and performance improvement of the IT incident management process," *IEEE Transactions on Network and Service Management*, vol. 7, no. 3, pp. 132–144, 2010.

[8] H. G. Cobb, "An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments," Naval Research Lab, Washington, D.C., USA, Tech. Rep. 6760 (NLR Memorandum), 1990.

[9] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art"," *Swarm and Evolutionary Computation*, vol. 6, no. 0, pp. 1–24, 2012.

[10] L. Foschini and M. Tortonesi, "Adaptive and business-driven service placement in federated Cloud computing environments," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 1245–1251.

[11] G. Grabarnik, M. Tortonesi, and L. Shwartz, "Business-Driven Optimization of Component Placement for Complex Services in Federated Clouds," in *Network Operations and Management Symposium (NOMS 2014), 2014 IEEE/IFIP*. IEEE, 2014, pp. 1–9.

[12] H. Moens, B. Hanssens, B. Dhoedt, and F. De Turck, "Hierarchical Network-Aware Placement of Service Oriented Applications in Clouds," in *Network Operations and Management Symposium (NOMS 2014), 2014 IEEE/IFIP*. IEEE, 2014, pp. 1–9.